Hierarchical Ensemble Classification: Towards the Classification of Data Collections that Feature Large Numbers of Class Labels

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Esra'a Ahmad Helael Alshdaifat

November 2015

Dedication

To my family, especially my parents and my son.

Abstract

In this thesis a number of hierarchical ensemble classification approaches are proposed as a solution to the multi-class classification problem. The central idea is that a more effective classification can be produced if a "coarse-grain" classification (directed at groups of classes) is first conducted followed by increasingly more "fine-grain" classifications. The Hierarchical ensemble classification model comprises a set of base classifiers held within the nodes of the hierarchy (one classifier per node). Nodes near the root hold classifiers designed to discriminate between groups of class labels while the leaves hold classifiers designed to distinguish between individual class labels. Two types of hierarchy (structures) are considered, Binary Tree (BT) hierarchies and Directed Acyclic Graph (DAG) hierarchies. With respect to the DAG structure, two alternative DAG structures to support the generation of the desired hierarchical ensemble classification model are considered: (i) *rooted* DAG, and (ii) *non-rooted* DAG. The main challenges are: (i) how best to distribute class labels between nodes within the hierarchy, (ii) how to address the "successive mis-classification" issue associated with hierarchical classification where if a mis-classication occurs early on in the process (near the root of the hierarchy) there is no possibility of rectifying this error later on in the process, and (iii) how best to determine the starting node within the *non-rooted* DAG approach. To address the first issue different techniques, based on the concepts of clustering, splitting, and combination, are proposed. To address the second and the third issues the idea is to utilise probability or confidence values associated with Naive Bayes and CARM classifiers respectively to dictate whether single or multiple paths should be followed at each hierarchy node, and to select the best starting DAG node with respect to the *non-rooted* DAG approach.

Keywords: Hierarchical Classification, Multi-class classification, Ensemble classification, Binary Tree (BT), Directed Acyclic Graph (DAG).

Acknowledgements

First and foremost, my deepest acknowledgment goes to my first supervisor Prof. Frans Coenen, for his constant support, constructive criticism, patience, research ideas and encouragement throughout the entire PhD program. Without his supervision and constant help the successful completion of this thesis would not have been possible. Honestly, it has been a great privilege to have worked with him. I would like to express my appreciation to my second supervisor Dr. Keith Duers for his support and encouragement during my PhD study. I am also thankful to my advisors Dr. Clare Dixon, Dr. Katie Atkinson, and Dr. Prudence Wong for their advices and valuable comments. I would like to extend my gratitude to the Hashemite University in Jordan for the generous financial support that has given me the opportunity to complete my PhD study. A special thanks also to my beloved family: my mother, father, sisters, and brothers, for their prayers, support and encouragement. Last but not least, I would like to express my sincere thanks to my husband for his support and encouragement throughout the entire PhD study.

Contents

Dedication i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Motivations . . . 2
- 1.3 Research Question and Issues . . . 3
- 1.4 Research Methodology . . . 5
- 1.5 Evaluation Data Sets and Criteria . . . 7
- 1.6 Research Contributions . . . 10
- 1.7 Organisation of the Thesis . . . 11
- 1.8 Publications . . . 12
- 1.9 Summary . . . 13

2 Literature Review 14
- 2.1 Introduction . . . 14
- 2.2 Using "Stand-alone" Classification Algorithms to Solve Multi-class Classification Problems . . . 15
- 2.3 Using a Collection of Binary Classifiers to Solve Multi-class Classification Problems . . . 17
- 2.4 Using Ensemble Classifiers to Solve Multi-class Classification Problems . . . 18
  - Part 1: Concurrent Ensemble Methods . . . 22
  - Part 2: Sequential Ensemble Methods . . . 23
  - Part 3: Binary Tree Based Hierarchical Ensemble Methods . . . 25
  - Part 4: DAG Based Hierarchical Ensemble Classification Methods 32
- 2.5 Comparison with Previous Work on Binary Tree Based Hierarchical Ensemble Methods . . . 35
- 2.6 Clustering . . . 35
- 2.7 Overview of Statistical Tests . . . 37
- 2.8 Terminology . . . 38
- 2.9 Summary . . . 39

3 The Binary Tree Hierarchical Classification Model 41

3.1 Introduction . . . 41
3.2 Binary Tree Hierarchical Model Generation . . . 42
3.3 Binary Tree Hierarchical Model Operation . . . 44
3.3.1 Single Path Strategy . . . 46
3.3.2 Multiple Path Strategy . . . 47
The Multiple Path Strategy Using CARM Classifiers at Nodes . . 48
The Multiple Path Strategy Using Naive Bayes Classifiers at Nodes 50
3.4 Experiments and Results . . . 52
3.4.1 Single Path Experiments and Results . . . 56
3.4.2 Multiple Path Experiments and Results . . . 65
Using Naive Bayesian Probability Values for Following Multiple Paths Within the Binary Tree Hierarchical Classification Model . . . 67
Using CARM Confidence Values for Following Multiple Paths Within the Binary Tree Hierarchical Classification Model . . . 69
Comparison Between Using Probability Values and Confidence Values for Following Multiple Paths Within the Binary Tree Hierarchy . . . 72
3.4.3 Comparison Between Single Path and Multiple Path Strategies . . 73
3.4.4 Comparison Between The Binary Tree Hierarchical Classification Model and Conventional models . . . 76
3.5 Summary . . . 81

4 Rooted Directed Acyclic Graph (*rooted* DAG) for Generating the Hierarchical Classification Model 89
4.1 Introduction . . . 89
4.2 Rooted DAG Generation . . . 90
4.3 Rooted DAG Operation . . . 91
4.3.1 Single Path Strategy . . . 93
4.3.2 Multiple Path Strategy . . . 94
4.4 Experiments and results . . . 98
4.4.1 Single Path Strategy Experiments and Results . . . 100
4.4.2 Multiple Path Strategy Experiments and Results . . . 101
4.4.3 Comparison Between Single and Multiple Path Strategies . . . 106
4.4.4 Comparison Between the Rooted DAG Ensemble Classification Model and Conventional models . . . 107
4.5 Summary . . . 114

5 Directed Acyclic Graph (DAG) Structure Based Hierarchical Classification Model 116
5.1 Introduction . . . 116
5.2 Non-Rooted DAG Generation . . . 118
5.3 Non-Rooted DAG Operation . . . 120
5.3.1 Single Path Strategy . . . 120
5.3.2 Multiple Path Strategy . . . 122
5.4 Experiments and Results . . . 125
5.4.1 Single Path Experiments and Results . . . 126
5.4.2 Multiple Path Experiments and Results . . . 129

5.4.3 Comparison Between Single Path and Multiple Path Strategies . . 132
5.4.4 Comparison Between Rooted and Non-rooted DAGs 135
5.5 Summary 139

6 The Directed Acyclic Graph (DAG) Hierarchical Classification Model with Breadth Pruning 142
6.1 Introduction 142
6.2 DAG Generation with the Application of Breadth Pruning 143
6.3 DAG Operation 145
6.4 Experiments and Results 146
6.4.1 Single Path Experiments and Results 147
6.4.2 Multiple Path Experiments and Results 148
6.4.3 Comparison Between Single Path and Multiple Path Strategies . . 149
6.4.4 Comparison Between Different DAG Models 151
6.5 Comparison Between DAG Based Hierarchical Classification and Binary Tree Based Hierarchical Classification 153
6.5.1 Comparison Between The DAG Ensemble Classification Model and Conventional models 156
6.6 Summary 159

7 Utilising Parallel Computing to Generate the Directed Acyclic Graph Classification Model 163
7.1 Introduction 163
7.2 Overview of Parallel Computing 164
7.2.1 Definition 164
7.2.2 Categorisation of Parallel Architecture 164
7.2.3 Parallel Programming Paradigms 165
7.2.4 Parallel Programming Languages 167
7.3 Utilising Parallel Processing to Generate and Operate the DAG Classification Model 167
7.3.1 Assigning Each DAG Node to a Process 168
7.3.2 Assigning Each DAG Level to a Process 169
7.3.3 Assigning Each DAG Level to a Group of Processes 171
7.4 Experiments and Results 174
7.4.1 Naive Byes rooted DAG Run Time Results 175
7.4.2 OVO SVM rooted DAG Results 176
7.5 Summary 179

8 Conclusion and Future Work 180
8.1 Summary 180
8.2 Main Findings and Contributions 182
8.3 Future Directions 187

Bibliography 188

Appendices **197**

A Evaluation data sets **198**

B AUC Computation **207**
B.1 Example One (100% Accurate Classifier) . . . 208
B.2 Example Two (Highly Unbalanced Data Set and One Class Does not Appear in the Test set) . . . 209

C Statistical Comparisons **212**
C.1 Introduction . . . 212
C.2 Binary Tree Hierarchical Classification Model Statistical Evaluation . . . 212
C.2.1 Comparative Evaluation using the Binary Tree Hierarchical Classification Model with the Single Path Strategy . . . 214
C.2.2 Comparative Evaluation using the Binary Tree Hierarchical Classification Model with the Multiple Path Strategy . . . 217
C.2.3 Comparison between Single and Multiple Path Strategies when using The Binary Tree Hierarchical Classification Model . . . 221
C.3 DAG Classification Models Evaluation . . . 222
C.3.1 Comparative Evaluation using the DAG Classification Models with the Single Path Strategy . . . 223
C.3.2 Comparative Evaluation using the DAG Classification Models with the Multiple Path Strategy . . . 223
C.3.3 Comparison between Single and Multiple Path Strategies using the DAG Classification Models . . . 225
C.3.4 Comparison Between the Different DAG Models . . . 230
C.4 Comparison Between DAG Based Hierarchical Classification and Binary Tree Based Hierarchical Classification . . . 231
C.5 Comparison Between the Hierarchical Classification Model and Conventional Ensemble Classification Models . . . 233
C.6 Summary . . . 235

D Determining the Best Threshold Value (σ) for Following Multiple Paths within the Binary Tree Hierarchical Classification Model **239**

E Following All Possible Paths Within Rooted DAG **254**

F Determining the Best Threshold Value With respect to the Rooted DAG Classification Model **256**

G Determining the Best Threshold Value With respect to the DAG Classification Model **264**

H Determining the Best Threshold Values With respect to the DAG Classification Model with Breadth Pruning **268**

Illustrations

List of Figures

1.1 Hierarchical classification structures, techniques, strategies, and mechanisms 5
1.2 ROC curve example 9

2.1 Binary Tree Hierarchy example. 26
2.2 Directed Binary Tree (DBT) example [72] 31
2.3 Decision Directed Acyclic Graph (DDAG) example [80] 33
2.4 Adaptive Directed Acyclic Graph (ADAG) example [55] 34

3.1 Binary Tree Hierarchy example. 42
3.2 Binary Tree hierarchical classification model, **(a)** using a single classifier at each node and **(b)** using a Bagging ensemble at each node. 43

4.1 Rooted DAG example 90

5.1 DAG example. 117

7.1 DAG example with parallel processing application 172

C.1 Hierarchical classification structures, strategies, techniques, and mechanisms 213
C.2 Results of Friedman test used to compare the different considered data distribution techniques with respect to Naive Bayes classification and the Single Path strategy 215
C.3 Results of Friedman test used to compare the different considered data distribution techniques with respect to Decision tree classification and Single Path strategy 215
C.4 Results of Friedman test used to compare the different considered data distribution techniques with respect to CARM classifier and Single Path strategy 216
C.5 Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) utilised to generate the Binary Tree classification model with respect to k-means data distribution technique and Single Path strategy 217
C.6 Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) utilised to generate the Binary Tree classification model with respect to data splitting technique and Single Path strategy 217
C.7 Results of Friedman test used to compare the different data distribution techniques with respect to Naive Bayes classification and the Multiple Path strategy 218
C.8 Results of Friedman test used to compare the different data distribution techniques with respect to CARM classification and the Multiple Path strategy 219

C.9 Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to the different data distribution techniques and the Multiple Path strategy . . . 220
C.10 Results of Friedman test used to compare the three alternative mechanisms to arrive at a final classification decision with respect to the Multiple Path strategy: (i) Voting, (ii) BIP, and (iii) NAP with respect to data splitting mechanism . . . 220
C.11 Results of Friedman test used to compare the three alternative mechanisms to arrive at a final classification decision with respect to the Multiple Path strategy: (i) Voting, (ii) BIP, and (iii) NAP with respect to K-means data distribution technique . . . 221
C.12 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the three considered data distribution techniques . . . 222
C.13 Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) with respect to the rooted DAG classification model and the Single Path strategy . . . 224
C.14 Results of Wilcoxon test used to compare the two considered classifiers with respect to: (i) All-level and (ii) Two-level DAGs, and the Single Path strategy 225
C.15 Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to the rooted DAG multiple path model . . . 226
C.16 Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to: (i) All-level, and (ii) Two-level DAG multiple path model . . . 226
C.17 Results of Fridman test used to compare the operation of the three alternative mechanisms used to arrive at a final classification when using the proposed Multiple Path strategies: (i) Voting, (ii) BIP, and (iii) NAP with respect to Naive Bayes classification and the rooted DAG approach . . . 227
C.18 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the rooted DAG model . . . 227
C.19 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the All-level DAG model . . . 228
C.20 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Two-level DAG model . . . 229
C.21 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Max-level DAG model . . . 229

C.22 Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Min-level DAG model . . . 230
C.23 Results of Friedman test used to compare the different DAG models with respect to the Single Path strategy . . . 231
C.24 Results of Friedman test used to compare the different DAG models with respect to the Multiple Path strategy . . . 232
C.25 Results of the Wilcoxon test used to compare the Binary Tree hierarchical classification Model and the DAG classification model with respect to the Single Path strategy . . . 233
C.26 Results of the Wilcoxon test used to compare the Binary Tree hierarchical classification Model and the DAG classification model with respect to the Multiple Path strategy . . . 234
C.27 Results of Friedman test used to compare the DAG classification model, stand-alone Naive Bayes classification and Bagging classification . . . 234
C.28 Results of Wilcoxon test used to compare the DAG classification Model and OVO SVM . . . 235

List of Tables

1.1 Data sets Characteristics . . . 8

2.1 Binary Tree hierarchical classification approaches . . . 27

3.1 Binary Tree hierarchical classification model variations . . . 55
3.2 Average Accuracy and AUC values obtained using Decision Tree classifiers and Bagging of Decision Trees classifiers at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting . . . 58
3.3 Run time results (in seconds) obtained using Decision Tree classifiers and Bagging of Decision Tree classifiers at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting . . . 59
3.4 Average Accuracy and AUC values obtained using Naive Bayes classifiers and Bagging of Naive Bayes classifiers at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting with respect to the Single Path Strategy . . . 60
3.5 Run time results (in seconds) obtained using Naive Bayes classifiers and Bagging of Naive Bayes classifiers at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting . . . 61

3.6 Average Accuracy and AUC values obtained using CARM classification and Bagging of CARM classification at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting with respect to the Single Path Strategy 63
3.7 Run time results (in seconds) obtained using CARM classification and Bagging of CARM classification at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 64
3.8 Average Accuracy and AUC values obtained using the Single Path strategy coupled with the three alternative classification algorithms for generating the node classifiers in the Binary Tree: (i) Decision Tree, (ii) Naive Bayes and (iii) CARM, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 66
3.9 Average Accuracy and AUC values obtained using the Multiple Path strategy coupled with the three alternative class label selection mechanisms: (i) BIP, (ii) NAP and (iii) Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 68
3.10 Run time results (in seconds) obtained using the Multiple Path strategy coupled with NAP with respect to Naive Bayes classification and the three considered data distribution techniques: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering 70
3.11 Average Accuracy and AUC values obtained using the Multiple Path strategy coupled with the two alternative class label selection mechanisms: (i) BIC, (ii) NAC and (iii) Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 71
3.12 Run time results (in seconds) obtained using the Multiple Path strategy coupled with NAP with respect to CARM classifiers and the three considered data distribution techniques: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering 72
3.13 Average Accuracy and AUC values obtained using Naive Bayes and CARM to generate Binary Tree hierarchical classification models coupled with the Multiple Path strategy (using NAP mechanism), with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 74
3.14 Average Accuracy and AUC values obtained using Naive Bayes to generate Binary Tree hierarchical classification models coupled with the Single and Multiple Path strategies, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 75

3.15 Average Accuracy and AUC values obtained using CARM to generate Binary Tree hierarchical classification models coupled with the Single and Multiple Path strategies, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting 77
3.16 Average accuracy and AUC results obtained when using: (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) Binary Tree hierarchies with decision trees at nodes (K-means&DT, DS&DT, and HC&DT) 79
3.17 Average accuracy and AUC results obtained when using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes and (iii) Binary Tree hierarchies with Naive Bayes classifiers at nodes (K-means&N, DS&N, and HC&N) 80
3.18 Average accuracy and AUC results obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM and (iii) Binary Tree hierarchies with CARM classifiers at nodes (K-means&CARM, DS&CARM, and HC&CARM) 82
3.19 Run time results (in seconds) obtained using (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) Binary Tree hierarchies with decision trees at nodes (K-means&DT, DS&DT, and HC&DT) 83
3.20 Run time results (in seconds) obtained using (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes and (iii) Binary Tree hierarchies with Naive Bayes classifier at nodes (K-means&N, DS&N, and HC&N) . . . 84
3.21 Run time results (in seconds) obtained using: (i) stand alone CARM classification, (ii) bagging of CARM and (iii) Binary Tree hierarchies with CARM classifiers at nodes (K-means&CARM, DS&CARM, and HC&CARM) 85

4.1 The optimal values for C and γ with respect to the fourteen considered evaluation datasets 99
4.2 Average Accuracy and AUC values obtained using Decision Tree, Naive Bayes and CARM to generate a rooted DAG classification model 100
4.3 Run time results (in seconds) obtained using Decision Tree, Naive Bayes, and CARM to generate a rooted DAG classification model 102
4.4 Comparison between: (i) NAP, (ii) BIP and (iii) Voting mechanisms for determining the final resulting class label with respect to rooted DAG 103
4.5 Average accuracy and average AUC results obtained using the two, three and all branches strategies when generating a rooted DAG (Naive Bayes) . . 104
4.6 Run time results (in seconds) obtained when using the two, three and all branch strategies when generating a rooted DAG (Naive Bayes) 105
4.7 Average Accuracy and AUC values obtained using Naive Bayes and CARM to generate rooted DAG classification models coupled with the two branch strategy 106

4.8 Average Accuracy and AUC results obtained using Naive Bayes coupled with either a Single or a Multiple Path strategy . . . 107
4.9 Average Accuracy and AUC results obtained using CARM coupled with either a single or a multiple path strategy . . . 108
4.10 Average accuracy and AUC results obtained when using: (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) DAG classification with decision trees at nodes . . . 109
4.11 Run time results (in seconds) obtained using (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) DAG classification with decision trees at nodes . . . 109
4.12 Average accuracy and AUC obtained when using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes classifiers and (iii) DAG classification with Naive Bayes classifiers at nodes . . . 110
4.13 Run time results (in seconds) obtained using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes classifiers and (iii) DAG classification with Naive Bayes classifiers at nodes . . . 111
4.14 Average Accuracy and AUC values obtained using Naive Bayes DAG coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier . . . 112
4.15 Run time results (in seconds) obtained using Naive Bayes DAG coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier . . . 112
4.16 Average accuracy and AUC results obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM classifiers and (iii) DAG classification with CARM classifiers at nodes . . . 113
4.17 Run time results (in seconds) obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM classifiers and (iii) DAG classification with CARM classifiers at nodes . . . 113

5.1 Average Accuracy and AUC values obtained using the Naive Bayes DAG model coupled with the Single Path strategy when using different numbers of levels in the DAG . . . 127
5.2 Average Accuracy and AUC values obtained using the CARM DAG model coupled with the Single Path strategy when using different numbers of levels in the DAG . . . 128
5.3 Comparison of average Accuracy and AUC values obtained using Naive Bayes DAGs and CARM DAGs coupled with the Single Path strategy . . . 129
5.4 Run time results (in seconds) obtained using Naive Bayes DAGs and CARM DAGs and the Single Path strategy . . . 129
5.5 Average Accuracy and AUC values obtained using Naive Bayes and CARM classifier generation with respect to Multiple Path DAGs . . . 131

5.6 Run time results (in seconds) obtained using Naive Bayes DAGs and CARM DAGs coupled with the Multiple Path strategy 132
5.7 Average Accuracy and AUC values obtained using Naive Bayes All-level DAGs coupled with either the Single Path or Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$) 133
5.8 Average Accuracy and AUC values obtained using Naive Bayes Two-level DAG coupled with either the Single Path or Multiple Path strategy ($\sigma = 0.5 \times 10^{-4}$) 134
5.9 Average Accuracy and AUC values obtained using CARM DAGs coupled with both the Single Path and Multiple Path strategies ($\sigma = 45$ and $\sigma = 40$ were used when following multiple paths with respect to the All-level and the Two-level DAGs respectively) 134
5.10 Average Accuracy and AUC values obtained using Naive Bayes *rooted* and *non-rooted* DAGs coupled with Single Path and Multiple Path strategies . . 136
5.11 Run time results (in seconds) obtained using Naive Bayes *rooted* and *non rooted* DAGs coupled with Single and Multiple Path strategies 137
5.12 Average Accuracy and AUC values obtained using CARM *rooted* and *non-rooted* DAGs coupled with Single Path and Multiple Path strategies 138
5.13 Run time results (in seconds) obtained using CARM rooted and non rooted DAGs coupled with Single and Multiple Path strategies 140

6.1 The best accuracy and AUC results obtained for each dataset using different α values with respect to Max-level and Min-level DAGs 148
6.2 Accuracy and AUC values obtained from following multiple paths within the DAGs (Min-level and Max-level) using a fixed α value ($\alpha = 0.40$) and the best α value with respect to each dataset 149
6.3 Average Accuracy and AUC results obtained using the Max-level DAG coupled with either a Single or a Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$) 150
6.4 Average Accuracy and AUC results obtained using Min-level DAG coupled with either a Single or a Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$) 151
6.5 Classification run time results (in seconds) obtained using the DAG model (Max-level and Min-level) 152
6.6 Accuracy and AUC results obtained using: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG, and (v) Min-level DAG coupled with Multiple Path strategy 154
6.7 Run time results (in seconds) obtained using DAGs based classification approaches: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG, and (v) Min-level DAG coupled with either Single Path or Multiple Path strategies 155
6.8 Average Accuracy and AUC values obtained using the Binary Tree classification model and the Min-level DAG classification model 156

6.9 Run time results (in seconds) obtained using the Binary Tree hierarchical model and the Min-level DAG classification model . . . 157
6.10 Average Accuracy and AUC values obtained using "Stand-alone" Naive Bayes classification, "Bagging" and the proposed *Min-level* DAG classification model . . . 158
6.11 Run time results (in seconds) obtained using "stand-alone" Naive Bayes classification, Bagging and the proposed *Min-level* DAG classification model 159
6.12 Average Accuracy and AUC values obtained using Naive Bayes DAG (Min-level DAG) coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier . . . 160
6.13 Run time results (in seconds) obtained using Naive Bayes DAG (Min-level DAG) coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier . . . 160

7.1 Run time results (in seconds) obtained using parallel processing to generate the Naive Bayes rooted DAG model compared with the Naive Bayes rooted DAG model generated without using parallel processing . . . 177
7.2 Average Accuracy and AUC values obtained using OVO SVM and the OVO SVM DAG . . . 178
7.3 Run time results (in seconds) obtained using OVO SVM and the OVO SVM DAG . . . 178

A.1 Waveform Characteristics . . . 198
A.2 Wine Characteristics . . . 199
A.3 Nursery Characteristics . . . 199
A.4 Heart Characteristics . . . 200
A.5 PageBlocks Characteristics . . . 200
A.6 Dermatology Characteristics . . . 201
A.7 Glass Characteristics . . . 201
A.8 Zoo Characteristics . . . 202
A.9 Ecoli Characteristics . . . 203
A.10 Led Characteristics . . . 203
A.11 PenDigits Characteristics . . . 204
A.12 Soybean Characteristics . . . 204
A.13 Chess KRvK Characteristics . . . 205
A.14 Letter Recognition Characteristics . . . 206

B.1 Truth values . . . 208
B.2 Prediction values . . . 208
B.3 MWW(1|2) . . . 208
B.4 MWW(2|1) . . . 208
B.5 MWW(1|3) . . . 209
B.6 MWW(3|1) . . . 209

B.7 MWW(2|3) . . . 209
B.8 MWW(3|2) . . . 209
B.9 Truth values . . . 210
B.10 Prediction values . . . 210
B.11 MWW(1|2) . . . 210
B.12 MWW(2|1) . . . 210
B.13 MWW(1|3) . . . 210
B.14 MWW(3|1) . . . 210
B.15 MWW(2|3) . . . 211
B.16 MWW(3|2) . . . 211

D.1 Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to k-mean data distribution technique, using a range of values for σ . . . 240
D.2 Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to data splitting technique, using a range of values for σ . . . 241
D.3 Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ . . . 242
D.4 Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to k-mean data distribution technique, using a range of values for σ . 243
D.5 Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to data splitting technique, using a range of values for σ . . . 244
D.6 Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ . . . 245
D.7 Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to k-mean data distribution technique, using a range of values for σ . . . 246
D.8 Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to data splitting technique, using a range of values for σ . . . 247
D.9 Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ . . . 248

D.10 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ 249
D.11 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ 249
D.12 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and *divisive* hierarchical clustering generated Binary Tree model, using a range of values for σ 250
D.13 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ 250
D.14 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ 251
D.15 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and *divisive* hierarchical clustering generated Binary Tree model, using a range of values for σ 251
D.16 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ 252
D.17 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ 252
D.18 Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and hierarchical clustering generated Binary Tree model, using a range of values for σ 253

F.1 Average Accuracy and AUC values obtained using different values for τ with respect to CARM in order to generate a rooted DAG classification model . . 257
F.2 Accuracy and AUC values produced when using the two branch strategy and the NAP mechanism with respect to a rooted DAG using a range of values for σ 258

F.3 Accuracy and AUC values produced when using the two branch strategy and the voting mechanism with respect to a rooted DAG using a range of values for σ . . . 259
F.4 Accuracy and AUC values produced when using the two branch strategy and the BIP mechanism with respect to a rooted DAG using a range of values for σ . . . 260
F.5 Accuracy and AUC values produced when using the three branch strategy with and the normalised accumulated weight mechanism with respect to a rooted DAG using a range of values for σ . . . 261
F.6 Accuracy and AUC values produced when using the all branch strategy with and the normalised accumulated weight mechanism with respect to a rooted DAG using a range of values for σ . . . 262
F.7 AUC values produced when using the two path strategy with respect to a CARM generated DAG model and a range of values for σ . . . 263

G.1 Results obtained using a range of σ values with respect to Naive Bayes classification, the multiple paths strategy and all-level DAGs . . . 265
G.2 Results obtained using a range of σ values with respect to Naive Bayes classification, the multiple path strategy and two-level DAGs . . . 266
G.3 Results obtained using a range of σ values with respect to CARM, the multiple paths strategy and all-level DAGs . . . 267
G.4 Results obtained using a range of σ values with respect to CARM, the multiple paths strategy and two-level DAGs . . . 267

H.1 Results obtained using a range of α values with respect to Max-levels DAG, using the single-path strategy . . . 269
H.2 Results obtained using a range of α values with respect to Min-levels DAG, using the single-path strategy . . . 270
H.3 Results obtained using a range of σ values for following multiple paths using the Max-levels DAG variation and $\alpha = 0.40$ with respect to breadth pruning 271
H.4 Results obtained using a range of σ values for following multiple paths using the Min-levels DAG variation and $\alpha = 0.40$ with respect to breadth pruning 272
H.5 Results obtained using a range of σ values for following multiple paths within the DAG using the Max-levels DAG variation and the best α value for each dataset with respect to breadth pruning . . . 273
H.6 Results obtained using a range of σ values for following multiple paths using the Min-levels DAG variation and the best α value for each dataset with respect to breadth pruning . . . 274

Chapter 1

Introduction

1.1 Overview

With the increasing availability of very large data collections the automated extraction (mining) of patterns from within such datasets is becoming increasingly challenging. This is true with respect to a variety of data mining processes including classification, especially where the classification task features a large number of class labels. Classification is a well established element of machine learning concerned with the creation of models, using pre-labelled "training" data, that can be used to allocate labels (classes) to previously unseen data. Classification can be viewed as a three-step process: (i) generation of the classifier using appropriately formatted "training" data, (ii) testing of the effectiveness of the generated classifier using test data and (iii) application of the classifier using previously unseen data. The first two steps are sometimes combined for experimental purposes.

The nature of the classification problem is characterised by two main factors: (i) the number of class labels that can be assigned to an example (single-label versus multi-label classification) and (ii) the number of classes from which the class labels may be drawn (binary versus multi-class classification). In single-label classification a classifier model is generated using a set of training examples where each example is associated with a single class label c taken from a set of disjoint class labels C ($|C| > 1$). If $|C| = 2$ we have a binary classification problem; if $|C| > 2$, we have a multi-class classification problem. The distinction between single-label and multi-label classification is that in multi-label classification the examples are each associated with a set of class labels Z, $Z \subseteq C$. In the work presented in this thesis we focus on the multi-class single-label classification problem where examples are associated with exactly one element of the set of class labels C. For simplicity, throughout this work, we will refer to this simply as "multi-class" classification. An issue with multi-class classification is that when $|C|$ is large the effectiveness of the classification tends to diminish. There has been extensive work directed at the generation of effective classifiers for multi-class classification problems [28, 58, 88, 94]. It is worth noting that, to date, no one classification model has been found to be superior to all others in terms of classification effectiveness [44].

Three main methodologies for addressing the multi-class classification problem can be identified: (i) using a single all-encompassing classifier, (ii) utilising a collection of binary classifiers and (iii) using an "ensemble" of classifiers. The Ensemble methodology is considered to be one of the most effective strategies to handle the multi-class problem [8, 27, 44, 47, 77, 79, 85, 109]. The ensemble model is a composite model comprised of a number of learners (classifiers), often referred to as base learners or weak learners, that "collaborate" to obtain a better classification performance than can be obtained from using a single "stand-alone" model. Classifiers making up an ensemble can be arranged in two main formats: (i) concurrent such as "Bagging" [12] and (ii) sequential such as "Boosting" [35]. In more recent work on ensemble classification, hierarchical arrangements of classifiers have been used [4, 20, 58, 60, 68, 100]. A commonly adopted structure is a binary tree constructed in either a bottom-up or top-down manner [11, 58]. The work presented in this thesis is directed at hierarchical ensemble classification.

The central idea espoused in this thesis is that a more effective classification can be produced if a "coarse-grain" classification (directed at groups of classes) is first conducted followed by increasingly more "fine-grain" classifications. To this end the generation and usage of a hierarchical ensemble classification models, that involve arranging the base classifiers in the form of a Binary Tree (BT) or Directed Acyclic Graph (DAG) structure, is proposed. Using these structures each node in the BT or DAG holds a classifier. Nodes near the root hold classifiers designed to discriminate between groups of class labels while the leaves hold classifiers designed to distinguish between individual class labels.

The remainder of this introductory chapter is organised as follows. Section 1.2 presents the motivations for the work presented in this thesis. Section 1.3 describes the main research question and the associated research issues to be addressed by the thesis. The adopted research methodology is presented in Section 1.4. Section 1.5 provides an overview of the data sets used to evaluate the proposed hierarchical ensemble classification approaches and a generic overview of the adopted evaluation metrics. Section 1.6 describes the contributions of the work presented. The organisation of the remainder of this thesis is presented in Section 1.7. Section 1.8 lists the publications resulting from the research presented in this thesis. Finally this chapter is concluded in Section 1.9 with a brief summary.

1.2 Motivations

From the foregoing, the work in this thesis is focused on using hierarchical ensemble classification to solve the multi-class classification problem.

The primary motivation for the work described in this thesis was a desire to provide a solution to a recognised problem in machine learning, the "multi-class" classification problem. It is generally simpler to construct a classifier for two mutually exclusive classes than for many (more than two) mutually exclusive classes. Multi-class classification is the problem of classifying examples into more than two classes. Given a training data

set D of the form (x_i, y_i), where $x_i \in D_n$ is the ith example in D, and $y_i \in \{1, \ldots, k\}$ is the ith class label in a given set C of such labels, the aim is to learn a model M such that $M(x_i) = y_i$ for new previously unseen examples. Multi-class classification is challenging because: (i) each class is represented by fewer examples in the training dataset than in the case of binary training data and (ii) a suitable subset of features that can be used to discriminate between large numbers of classes (more than two) is often difficult to identify.

Using ensembles of classifiers arranged in a hierarchical form is expected to provide an effective classification with respect to the multi-class classification problems for two reasons: (i) the established observation that ensemble methods tend to improve classification performance [8, 27, 44, 47, 77, 79, 85, 109] and (ii) dealing with smaller subsets of class labels at each node might produce better results.

In addition to the above motivations, the work described in this thesis was also motivated by two further challenges specific to the operation of hierarchical ensemble classification models. The first was how best to organise (group) the class labels at nodes so as to produce a hierarchy that generates the most effective classification. The second, and one which (to the best knowledge of the author) has not been addressed previously, was the "successive mis-classification" issue associated with hierarchical classification models. In other words, how to deal with the issue that if an example is mis-classified early on in the process (near the root of the hierarchy) it will continue to be mis-classified at deeper levels of the hierarchy, regardless of the classifications proposed at lower level nodes and the final leaf nodes.

From the forgoing, the motivation for the work described in this thesis can be summarised as follows:

1. A desire to provide a more effective form of classification for multi-class classification problems, especially in the case of datasets that feature a large number of class labels.

2. The expectation that the hierarchical classification model will produce an effective classification with respect to the multi-class classification problems.

3. A desire to address the "successive mis-classification" issue associated with hierarchical classification.

4. The necessity of a comprehensive study concerning a recent form of ensemble classification; namely hierarchical ensemble classification.

1.3 Research Question and Issues

Given the motivations presented in the foregoing section, the main research question to be addressed by this thesis was:

"What are the most appropriate mechanisms that can be employed to generate effective hierarchical classification models?"

In order to answer this research question, the resolution of a number of subsidiary research questions were required. These questions can be summarised as follows:

1. Is a hierarchical classifier best arranged using a Binary Tree structure, or is it better to adopt a Directed Acyclic Graph (DAG), to effectively classify data collections that feature a large number of class labels?

2. How can the nodes in a hierarchical classifier best be connected to achieve an effective classification?

3. Can a better classification accuracy be achieved by following more than one path within the hierarchy? And if so how do we decide which paths to follow?

4. Following on from (3) above, when adopting a multiple path strategy, how do we combine a number of possibly contradictory final classifications to provide a single end classification?

5. Following on from (3) and (4) above, will using a multiple path serve to address the "successive mis-classification" issue associated with hierarchical ensemble classification models?

6. What is the best way of dividing up a given set of class labels between nodes (in a Binary Tree hierarchy or DAG)? Previous research on hierarchical classification, which is mostly directed at the use of Binary Tree structures, proposed some techniques to distribute class labels between nodes within the hierarchy. However, to the best knowledge of the author, no previous work has provided a comparative study of these techniques. In another words, no recommendation for the "best" data distribution technique has been proposed.

7. What is the most appropriate classification algorithm to be held at individual nodes? In addition to the efficiency and effectiveness concerns usually used to evaluate classification algorithm, a further consideration is the support that individual classification algorithms provide with respect to any multiple path strategy that might be adopted.

8. Is it indeed the case that Binary Tree hierarchical classifiers and/or DAG classifiers can be more effectively used (than when using alternative techniques) to classify data collections that feature a large number of class labels?

9. **What is the most effective classification model to be used for classifying a new data set given some characteristics regarding the new data set such as: number of examples, number of classes and skewness?**

The specific objectives of the research were thus to find answers to the above research question and associated subsidiary questions.

1.4 Research Methodology

The adopted research methodology was to consider a series of techniques to generate classification hierarchies starting with simple Binary Tree structures and moving on to more complex DAG structures. Regardless of the structure of the desired hierarchies a key issue was how class labels are to be assigned to nodes and how the nodes are to be connected. Several techniques were considered to achieve this including: splitting, clustering and combination techniques. For each structure, how best to follow several paths in the hierarchy was also a consideration. Figure 1.1 presents the different structures, strategies, techniques, and mechanisms that have been considered in this thesis.

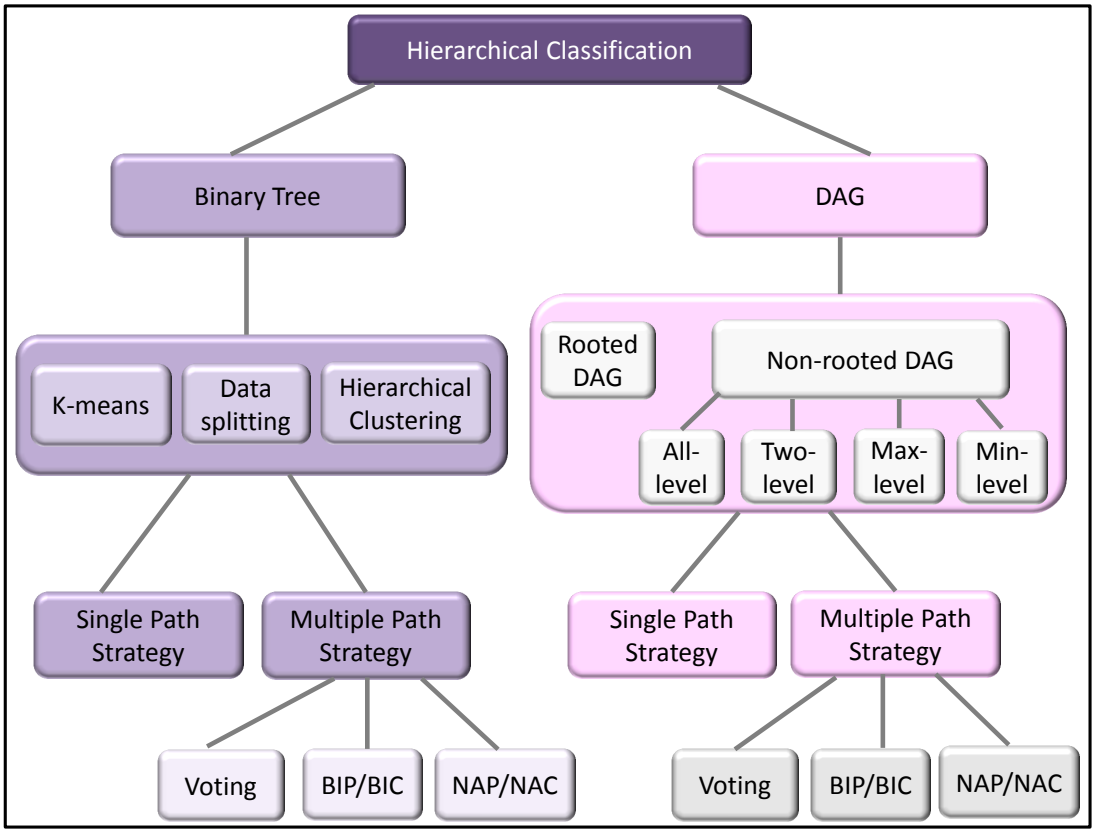


FIGURE 1.1: Hierarchical classification structures, techniques, strategies, and mechanisms

The first investigation conducted was directed at the use of a Binary Tree structure to generate the desired hierarchical classification model. To generate the Binary Tree hierarchy three different distribution techniques were proposed in order to divide the data during the hierarchy generation process (founded on ideas concerned with clustering and splitting techniques): (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering (as shown in Figure 1.1). The use of two different styles of classifier at each

hierarchy node was also proposed: (i) single "stand-alone" classifiers and (ii) "bagging" ensemble classifiers. Three alternative classification algorithms were considered: (i) Decision tree, (ii) Naive Bayes and (iii) Classification Association Rule Mining (CARM).

Two classification strategies were proposed: (i) "Single-Path" and (ii) "Multiple-Path" (again as shown in Figure 1.1). In the first case the strategy was to select the class at the leaf node label by following a "single path" within the hierarchy from the root node to the leaf node. The second strategy allowed for more than one path to be followed. This second strategy was proposed to address the "successive mis-classification" problem identified earlier in this chapter. In the case of the Multiple Path strategy this was specifically designed to operate using Naive Bayes or CARM classification, which feature probability or confidence values that can be used to determine whether single or multiple paths should be followed at each hierarchy node. In the case where more than one path is followed, it was anticipated that a number of alternative class labels would result, three alternatives for arriving at a final decision were proposed (as shown in Figure 1.1): (i) a voting mechanism; (ii) selecting the class label associated with the leaf node that features the highest probability (confidence) a measure referred as the Best Individual Probability or Confidence (BIP/BIC) measure; and (iii) taking into consideration the probability (confidence) values identified along the path back to the root node to produce an *accumulated value*, a measure referred to as the Normalised Accumulated Probability or Confidence (NAP/NAC) measure.

The second structure to be investigated was the use of DAGs to generate the desired hierarchical classification model. With reference to Figure 1.1 two alternative DAG hierarchical classification structures were proposed: (i) *rooted* DAG, and (ii) *non-rooted* DAG. To generate the DAG classification model "combination techniques" were proposed to distribute (organise) the class labels between nodes within the DAG. Again, as in the case of the proposed Binary Tree hierarchies, three alternative classification algorithms were considered: (i) Decision tree, (ii) Naive Bayes and (iii) Classification Association Rule Mining (CARM). Two classification strategies were again considered: (i) "Single-Path" and (ii) "Multiple-Path" together with, in the later case, the three alternatives for arriving at a final classification decision as used with respect to the binary tree structure investigated earlier: (i) Voting, (ii) BIP/BIC and (iii) NAP/NAC. The *non-rooted* DAG models were found to perform well, however, in order to improve the performance (effectiveness, efficiency, and scalability) of the *non-rooted* DAG model two forms of pruning were considered, depth and breadth pruning.

It is worth to noting here the reasons behind choosing Decision tree, Naive Bayes and CARM classification algorithms to generate the node classifiers:

1. A requirement for multi-class classification algorithms in the case of the DAG models.

2. The expectation that a better classification accuracy would be obtained when multi-class classification algorithms were utilised.

3. The fact that with respect to Naive Bayes and CARM classifiers, probability and confidence values could be utilised to: (i) support mechanisms for following multiple paths within a hierarchy and (ii) to determine the "start node" (root node) from which a classification process should best commence, with respect to the *non-rooted* DAGs.

4. A desire to use a consistent set of classification algorithms with respect to the evaluation of the different hierarchical classification models suggested.

In addition utilising parallel computing to generate and operate the proposed *rooted* DAG hierarchical classification model was considered. The conjecture here was that this would generate a more efficient and effective DAG. It was suggested that the latter can be realised if more effective classifiers, such as SVM classifiers, are used at each DAG node. However, "stand-alone" SVM classifiers could not be used in the context of the DAG model because they are essentially binary classifier. To address this issue OVO SVM was used at each DAG node. Consequently, the resulting model would be a form of ensemble of ensembles which might improve the classification effectiveness.

1.5 Evaluation Data Sets and Criteria

This section provides an overview of the data sets used to evaluate the proposed hierarchical ensemble classification approaches. A generic overview of the adopted evaluation metrics used with respect to the evaluation data sets is also presented.

In the context of the evaluation reported on later in this thesis, fourteen different data sets (with different numbers of class labels) were used taken from the UCI machine learning repository [63]. These were processed using the LUCS-KDD-DN data preprocessing software system [23]. The general characteristics of the data sets are provided in Table 1.1. The last column refers to the skewness of the data sets. More specifically, given a data set with N class labels if the percentage of the occurrences of each class is $100/N$ the data set is perfectly balanced. While if one or more class labels differ significantly from other classes, then the data set is skewed or unbalanced [38]. The distribution of each class label with respect to each data set, and further information about the evaluation data sets, is provided in Appendix A.

In order to evaluate the effectiveness of the proposed hierarchical classification approaches, average accuracy and average AUC (Area Under the receiver operating Curve) were used. To describe the calculation of these metrics in further detail it is first necessary to present some terminology in the context of binary classifiers:

1. True Positives (TP): The number of positive examples that are correctly labelled as positive.

2. True Negatives (TN): The number of negative examples that are correctly labelled as negative.

TABLE 1.1: Data sets Characteristics

Dataset	Number of Classes	Number of Examples	Number of Attributes	Missing Values	Balanced/ Skewness
WaveForm	3	5000	21	0	Balanced
Wine	3	178	13	0	Balanced
Nursery	5	12960	8	0	Skewed
Heart	5	297	13	6	Skewed
PageBlocks	5	5473	10	0	Skewed
Dermatology	6	358	12	8	Skewed
Glass	7	214	10	0	Skewed
Zoo	7	101	16	0	Skewed
Ecoli	8	336	7	0	Skewed
Led	10	3200	7	0	Balanced
PenDigits	10	10992	16	0	Balanced
Soybean	15	562	35	0	Skewed
Chess KRvK	18	28056	6	0	Skewed
Letter Recognition	26	20000	16	0	Balanced

3. False Positives (FP): The number of negative examples that are incorrectly labelled as positive.

4. False Negatives (FN): The number of positive examples that are incorrectly labelled as negative.

Given the above, the **accuracy** of a classifier, the percentage indicating the number of examples that are correctly classified, is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

More simply the accuracy of a given classifier is the number of examples that are correctly classified divided by the total number of examples in a given test set.

From the foregoing, accuracy is a straightforward and easy to calculate measure. However, the accuracy measure does not take into consideration the skewness of the data (class priors). Consequently, it is preferable to consider an alternative evaluation measure that does take skewness into account. Within the data mining community the most commonly used metric in this context is the Area Under the ROC Curve (AUC) metric [52]. AUC takes skewness into account, and as a result it is considered to be a more informative measure, especially when considering very skewed data sets (as the case of some of the work presented in this thesis).

To fully understand the AUC evaluation measure, it is first necessary to understand what a ROC Curve is. A Receiver Operating Characteristic (ROC) Curve is a visual representation of the trade-off between the *true positive rate*[1] (Recall/Sensitivity) and the *false positive rate*[2] (1-Specificity). Figure 1.2 shows three ROC curves for three different classifiers. The diagonal line indicates random guessing. A ROC curve located above the diagonal line indicates that the performance is better than a guess, while a

[1] $TPR = \frac{TP}{TP+FN}$
[2] $FPR = \frac{FP}{TN+FP}$

ROC curve located below the diagonal line indicates that the performance is worse than guessing. A best performance is indicated by a ROC curve located close to the top left hand corner of the plot. Thus, with respect to the ROC curves presented in Figure 1.2 the yellow curve is the best. Rather than considering the individual curves it is simpler to consider the area under the ROC curve, hence the AUC measure. An AUC of 1 indicates a perfect performance; an AUC of 0.5 indicates guessing; an AUC of less than 0.5 indicates a performance that is worse than guessing. A detailed example of the AUC calculation is provided in Appendix B.

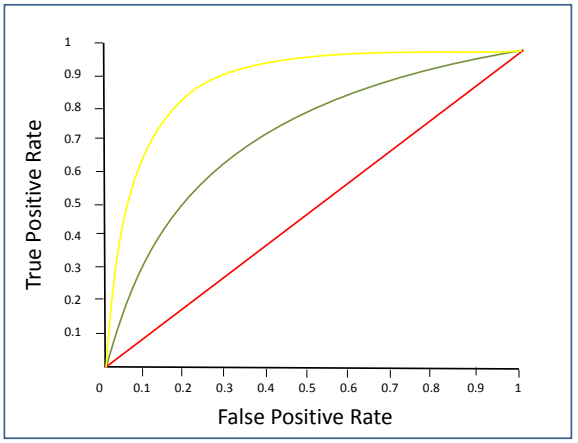


FIGURE 1.2: ROC curve example

In addition to classification effectiveness, efficiency was also considered with respect to the work presented in this thesis. Efficiency was measured according to run times, both hierarchical ensemble generation time and classification time. Note here that all experiments were conducted using a 2.7 GHz Intel Core i5 with 16 GB 1333 MHz DDR3 memory, running OS X 10.9.2 (13C64).

In order to obtain more accurate results average accuracies and average AUC values were obtained using the Tenfold Cross Validation (TCV) method. Using TCV a given data set D is partitioned into ten disjoint partitions, $d1, d2, \ldots, d10$, each of approximately equal size. Training is performed ten times each time using a different tenth as the test set and the remaining nine tenths as the training set. On each iteration accuracy and AUC was measured and at the end of process the overall accuracy and AUC were calculated.

It should be noted here that an issue arises when considering TCV with respect to the AUC value for some *highly* unbalanced data sets. More specifically, assuming a highly unbalanced data set that feature less than ten examples of a specific class, dividing the data set into ten folds results in some folds without any examples from that class. During the testing stage, the classifier will not be actually evaluated against that class for some test folds (the folds that do not include any example of that class). However the AUC calculations assume the complete number of class labels. Consequently a low AUC values will be produced. With respect to the work presented in this thesis, this

case was found for four data sets: (i) Nursery, (ii) Glass, (iii) Zoo and (iv) Ecoli. An example of such case is presented in Appendix A.

The effectiveness of the proposed models were evaluated by comparing with more conventional existing models (stand-alone multi-class classifiers, collection of binary classifiers and ensemble models). To determine whether the results obtained were statistically significant a precise and comprehensive statistical analysis of the results was conducted using the Wilcoxon signed rank test for comparing two classification models, and the Friedman test (coupled with a Nemenyi post-hoc test where appropriate) for comparing several classification models (more than two).

1.6 Research Contributions

The main contributions of the research presented in this thesis can be summarised as follows:

1. A set of alternative techniques to distribute class labels between nodes within a Binary Tree hierarchy. With respect to the existing work on Binary Tree hierarchies, it should be noted that the most frequently used methods for dividing classes between nodes do not allow overlapping between the class groups. In this work both overlapping and non-overlapping techniques were considered. The conjecture of allowing overlapping was that this would mitigate against the early mis-classification issue.

2. An evaluation of the use of a number of alternative classification algorithms, to generate node classifiers within a Binary Tree hierarchy. Note that existing work on Binary Tree hierarchies has mainly utilised binary classification algorithms such as SVM.

3. An "ensemble of ensembles" approach with respect to Binary Tree hierarchies. More specifically, using Bagging ensembles at each node within a binary tree hierarchy.

4. A Multiple Path strategy, which allows for more than one path to be followed within a hierarchy during the classification stage. This strategy is completely novel and it was proposed to address the "successive mis-classification" issue associated with hierarchical classification. Note here that this strategy was considered with respect to both the proposed Binary Tree and DAG hierarchies.

5. Three alternative mechanisms (Voting, BIP/BIC and NAP/NAC) for arriving at a final classification decision with respect to the Multiple Path strategy. The aim was to address the issue in the case where more than one path is followed where we end up with a number of alternative candidate class labels.

6. A unique *rooted* DAG structure for hierarchical multi-class classification.

7. A novel *non-rooted* DAG structure for hierarchical multi-class classification.

8. A novel mechanism for applying breadth pruning to the *non-rooted* DAG structure. The conjecture here was that this would improve the effectiveness and efficiency of the DAG classification model, because "weak" classifiers would be pruned.

9. A comprehensive study and statistical analysis of the proposed hierarchical ensemble classification models to identify the "best" structure, classification algorithm, data distribution technique and classification strategy to be adopted in order to obtain an effective and efficient hierarchical classification model.

10. Utilising parallel computing to generate and operate the proposed *rooted* DAG hierarchical classification model. The conjecture here was that this would generate a more efficient and effective DAG classification model that could be directed at even larger numbers of class labels.

1.7 Organisation of the Thesis

The remainder of this thesis is organised as follows:

Chapter 2 provides a review of the previous work that is of relevance with respect to the work presented in this thesis.

Chapter 3 presents the hierarchical ensemble classification model for multi-class classification founded on a Binary Tree (BT) structure.

Chapter 4 describes the nature of the proposed *rooted* DAG hierarchical ensemble classification model.

Chapter 5 presents the *non-rooted* DAG structure, rather than a *rooted* DAG structure, to generate the desired hierarchical classification model. The chapter also considers the application of *depth pruning* with respect to the *non-rooted* (DAG) structure.

Chapter 6 considers the application of *breadth pruning* with respect to the *non-rooted* (DAG) hierarchical ensemble classification model. The chapter also presents an evaluation of the application of both *depth pruning* and *breadth pruning* with respect to the *non-rooted* (DAG) structure.

Chapter 7 considers using parallel computing to generate the *rooted* DAG hierarchical classification model.

Chapter 8 begins by presenting some conclusions, then lists the main findings of the work presented in this thesis, and then presents some potential directions for future work.

1.8 Publications

Five papers, three published, one waiting publication, and one presented for refereeing have arisen out of the work presented in this thesis and these are listed and summarised in this section.

1. Journal Papers:

   (a) **Esra'a Alshdaifat, Frans Coenen, and Keith Dures. *The Directed Acyclic Graph (DAG) Ensemble Classification Model: An Alternative Architecture for Hierarchical Classification.* Submitted for refereeing to the International Journal of Data Warehousing and Mining (IJDWM).** This paper summarises the work presented in this thesis. More specifically, the paper proposes the two alternative DAG structures to support the generation of the desired DAG hierarchical classification approach: (i) *rooted* DAG and (ii) *non-rooted* DAG. The paper also presents a comparison between the Binary Tree and DAG structures to generate the hierarchical classification model. A comparison between the hierarchical classification and the well-established conventional models for multi-class classification was also included in this paper. The work in this paper is included in Chapters 5, 7 and 8.

2. Conference Papers:

   (a) **Esra'a Alshdaifat, Frans Coenen, and Keith Dures. *Hierarchical Single Label Classification: An Alternative Approach.* In Max Bramer and Miltos Petridis, editors, the thirty-third BCS SGAI International Conference on Artificial Intelligence (BCS SGAI 2013), pages 39-52. Springer, 2013.** This paper presents the use of the Binary Tree structure for use with the desired hierarchical classification model. In this paper a comparison between two different styles of classification at each hierarchy node was also considered: (i) single "stand-alone" classification and (ii) "bagging" ensemble classification. A comparison between the three different techniques for identifying the classes covered by nodes was also included: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering. The work presented in this paper acted as the foundation for the work described in Chapter 4 of this thesis.

   (b) **Esra'a Alshdaifat, Frans Coenen, and Keith Dures. *Hierarchical Classification for Solving Multi-class Problems: A New Approach Using Naive Bayesian Classification.* In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar R. Zaiane, Min Yao, and Wei Wang, editors, the 9th International Conference on Advanced Data Mining and Applications (ADMA 2013) (1), Lecture Notes in Computer**

Science, volume 8346, pages 493-504. Springer, 2013. This paper was the first to introduce the idea of following multiple paths within a Binary Tree ensemble hierarchy. The paper presented a comparison between the Single and Multiple Path strategies with respect to Naive Bayes classification. The paper also considered two mechanisms for arriving at a final classification decision with respect to the Multiple Path strategy. The majority of the content of this paper is included in Chapter 4 of this thesis.

(c) **Esra'a Alshdaifat, Frans Coenen, and Keith Dures. *A Multi-path Strategy for Hierarchical Ensemble Classification.* In Petra Perner, editor, the 10th Machine Learning and Data Mining in Pattern Recognition (MLDM 2014), Lecture Notes in Computer Science, volume 8556, pages 198-212. Springer, 2014.** This paper proposed utilising the confidence values associated with CARM classifiers for following multiple paths within a Binary Tree hierarchical classification model. A comparison was presented between the operation of: (i) the Single and Multiple Path strategies with respect to CARM classification, (ii) two mechanisms for arriving at a final classification decision in context of the Multiple Path strategy, (iii) the three different techniques to distribute class labels between nodes within the hierarchy and (iv) the Binary Tree hierarchical classification model and conventional models for multi-class classification. The majority of the content of this paper was included in Chapter 4.

(d) **Esra'a Alshdaifat, Frans Coenen, and Keith Dures. *Directed Acyclic Graphs for Multi-Class Classification.* Proceeding AI 2015, Springer, in press**. This paper proposed the *non-rooted* DAG classification model for multi-class hierarchical ensemble classification. The paper also presented the application of *breadth and depth* pruning with respect to the proposed *non-rooted* DAG classification model. The work presented in this paper is incorporated into Chapter 7 of this thesis.

1.9 Summary

This introductory chapter has presented a general overview and a background to the research described in this thesis. The motivation for the research, the research question together with the associated issues to be addressed, the adopted research methodology, evaluation data sets and criteria, and the contributions of the research have all been presented. The main objective of the research is to utilise hierarchical ensemble classification to provide a more effective classification mechanism for multi-class classification, especially in the context of datasets that feature a large number of class labels. The next chapter (Chapter 2) provides a literature review aimed at providing the necessary background to the work presented in this thesis.

Chapter 2

Literature Review

2.1 Introduction

As noted earlier in the introduction to this thesis, single-label classification (as opposed to multi-label classification) is concerned with the learning of classifiers, using a set of training examples, where each example is associated with a single class label c taken from a set of disjoint class labels C. If $|C| = 2$ we have a simple "binary" classification problem, if $|C| > 2$ it is referred to as a "multi-class" classification problem. In comparison with single-label classification, multi-label classification is more general, since it allows one example to have more than one label simultaneously [82]. Multi-label classification methods have been increasingly used in modern applications such as: (i) music categorisation where the same song can belong to several different classes, (ii) semantic classification where (say) an image can be classified as belonging to a number of classes, (iii) text categorisation where a document can belong to more than one conceptual class and (iv) medical diagnosis where a number of diagnosis (classes) may be applicable (further examples can be found in [96]). However, the majority of real-world classification problems tend to involve single-labels. Examples of single-label classification applications include: (i) social classification such as in the case of the UCI Nursery data set [63], (ii) medical diagnoses such as predicting the absence or presence of some condition and (iii) biology applications such as the prediction of protein localisation sites. Many examples of single class-label data sets can be found in the UCI machine learning data repository [63].

In the context of the work described in this thesis, as also noted in the introduction to this thesis, the focus is on single-label multi-class classification problem where examples are associated with exactly one element of the set of class labels C. We refer to this simply as "multi-class" classification. An issue with multi-class classification is that when $|C|$ is large the effectiveness of the classification tends to degrade. It is widely accepted that multi-class problems can be solved in three ways: (i) using "stand-alone" classification algorithms, (ii) using a set of binary classifiers and (iii) using ensemble classifiers arranged in some specific form. Basic surveys of the fundamental techniques used to solve multi-class classification problems can be found in [3, 71]. However, these two

published surveys only give an overview of small number of techniques that can be used to address multi class-classification problems. In this chapter, a more comprehensive overview is presented.

The section is organised as follows: Section 2.2 considers the classification algorithms that can be directly used to address multi-class classification problems. Section 2.3 reviews using binary classifiers to solve the multi-class problem. Section 2.4 then considers the usage of ensemble methodologies in the context of the multi-class classification problem. Section 2.5 provides a comparison with the previous work on binary tree based hierarchical ensemble methods. This is followed in Section 2.6 with a generic overview of clustering algorithms, the reason behind the inclusion of this section is that clustering algorithms were utilised to distribute classes between nodes within the proposed ensemble hierarchies. Section 2.7 provides a general overview of the statistical tests used latter in this thesis to compare the different classification models. Section 2.8 explains some terminology used latter in this thesis. Finally, a summary of this chapter is presented in Section 2.9.

2.2 Using "Stand-alone" Classification Algorithms to Solve Multi-class Classification Problems

Some classification algorithms are specifically designed to address binary classification, for example support vector machines [98]. However, such algorithms can be adapted to handle multi-class classification by building sequence of binary classifiers. Other classification algorithms can directly handle any number of class labels; examples include: decision tree classifiers [83], Classification Association Rule Mining (CARM) [24], Neural Networks [108], k-Nearest Neighbors[9], and Bayesian classification [61]. Among these, decision tree algorithms are of interest with respect to the work described in this thesis because it can be argued that the proposed hierarchies have some similarity with the concept of decision trees. Decision trees have a number of advantages with respect to some other classification techniques: (i) they can be constructed relatively quickly, (ii) they are easy to understand (and modify), (iii) the tree can be expressed as a set of "decision rules" (which is of benefit with respect to some applications) and (iv) the accuracy of decision tree classifiers is comparable or superior to other classification models [64]. Decision trees are constructed by inducing a "split" in the training data according to the values associated with the available attributes. The splitting is frequently undertaken according to "Information Gain" [83], "Gini Gain" [14] or "Gain Ratio" [83]. No one attribute selection measure has been found to be superior to others, most measures give quite good results [44]. Each leaf node of a decision tree holds a class label. A new example is classified by following a path from the root node to a leaf node, the class held at the identified leaf node is then considered to be the class label for the example [44]. Amongst the most frequently quoted decision tree generation algorithms are: ID3 [83],

C4.5 [84] and CART [14]. A comprehensive survey of work on decision tree classifiers can be found in [74].

Bayesian classification algorithms are also of interest with respect to the work described in this research because of the Bayesian probability values generated; as will become clear later in this thesis, these probability values were used so as to enable more than one branch in a hierarchy to be followed. Bayesian classification is based on Bayes' theorem [44]. A simple Bayesian classifier is known as a Naive Bayesian classifier. The main assumption of Naive Bayes classification is that the effect of an attribute value on a given class is independent of the values of other attributes [44]. This is the well-known "class conditional independence" assumption which is adopted to simplify the computations [44]. By Bayes' theorem the probability that example E is of class C_i is given by [44]:

$$P(C_i|E) = \frac{P(E|C_i)P(C_i)}{P(E)} \tag{2.1}$$

where: (i) $P(C_i|E)$ is the posterior probability, or a posteriori probability, of C_i conditioned on E (the probability that example E is of class C_i), (ii) $P(E|C_i)$ is the posterior probability of E conditioned on C_i and (iii) $P(C_i)$ is the prior probability of C_i and (iv) $P(E)$ is the prior probability of E.

Because $P(E)$ is fixed for all classes, we only need to maximise $P(E|C_i)P(C_i)$. Naive Bayes is a highly effective and straightforward form of classification. Consequently it is often used as a baseline standard by which other classifiers can be measured [39]. Various comparative studies, with respect to decision tree and neural network classifiers, have found the operation of Naive Bayes to be comparable [17, 30, 59]. In addition, Naive Bayes classifiers have produced high accuracy and speed with respect to large sized data sets [44].

Classification Association Rule Mining (CARM) algorithms are also of interest with respect to the work described in this thesis. The significance is that CARM incorporates the concept of confidence values which in turn, it is argued later in this thesis, can be used to determine the most appropriate paths through a hierarchical ensemble classification model. CARM integrates Association Rule Mining (ARM) and classification. CARM algorithms work by applying an association rule mining style algorithm, such as Apriori [1] or FPgrowth [45], to produce classification rules from pre-labelled training data [24] according to: (i) a user defined support (frequency) threshold[1] and (ii) a user defined confidence threshold[2]. The aim is to generate association rules that have only a single class label in the consequent. The generated association rules are referred to as Classification Association Rules (CARs) [62], which collectively form the desired classifier. CARM algorithms can be categorised according to how the pruning of low confidence CARs is performed [24]: (i) two stage or (ii) integrated. In the two

[1]The support of a rule describes the number of examples in the training data where the rule antecedent and consequent occur [24].

[2]The confidence of a rule is the ratio of its support to the support for its antecedent [24].

stage approach all CARs are generated in the first stage and pruned in the second stage. Examples of this approach include Classification based on Multiple Association Rules (CMAR) [62], and Classification Based on Associations (CBA) [65]. Using integrated algorithms the classifier generation is accomplished in a single processing step encompassing both rule generation and pruning. Examples of this latter approach include Classification based on Predictive Association Rules (CPAR) [107] and Total From Partial Classification (TFPC) [24]. Although experimental work has shown that CARM improves classification accuracy when compared to stand-alone classification algorithms such as C4.5 [62, 65], two main drawbacks can be identified: (i) high processing overhead, due to the generation of large number of association rules and (ii) overfitting as the result of the application of the confidence-based rule evaluation measure [107].

2.3 Using a Collection of Binary Classifiers to Solve Multi-class Classification Problems

With the availability of many robust binary classification algorithms; the multi-class classification problems can be addressed by utilising a collection of binary classifiers; the multi-class problem is thus decomposed into a number of binary classification sub-problems that can be resolved using binary classifiers. We can, from the literature, identify three commonly referenced methods of using binary classifiers to solve multi-class problems: (i) One-Versus-All (OVA) [88] (ii) One-Versus-One (OVO) [94] and Error-Correcting-Output-Codes (ECOC) [28].

Commencing with the simplest method of using binary classifiers to solve multi-class problem, which involves training N binary classifiers to handle a N class problem. Each classifier is trained to discriminate the examples in a single class from the examples in all remaining classes. This method is thus the OVA method [88]. The ideal case, for classifying a new example, is that one classifier generates output 1 and all the remaining classifiers generate output -1; as a result class with the output 1 will be assigned to the new example. However, the following cases might be raised: (i) more than one classifier assigns the example to its class or (ii) none of the classifiers assign the example to its class. In order to address these issues a winner-take-all (WTA) strategy [48] can be adopted, where a real-valued function is assigned to each class in order to determine the class membership. For classifying a new example, the classifier that generate the maximum output is considered the "winner".

With respect to OVO [94] (also called All-Versus-All (AVA) [44]), a classifier is trained for every possible pair of classes. Consequently, if we have N classes, then $(N(N-1)/2)$ classifiers are required to be trained. For classifying a new unseen example, each classifier "votes", and the class with the maximum number of votes is assigned to the new example (Max-wins [36]). However, if more than one class is assigned with the same number of votes, a number of "candidate classes" will be available. Then, the final classification result will be selected randomly from the set of "candidate classes". The

main drawback is the large number of classifiers required, although reports from the literature suggest that OVO tends to be superior to OVA [44].

The ECOC method operates by changing the definition of the class a single classifier has to learn [28]. The original motivation for ECOC was to improve the operation of binary classification but it can be applied with respect to the multi-class problem. A unique binary string "codeword" of length n is assigned to each class. For each bit position of the binary strings a classifier is trained, resulting in n binary classifiers. For classifying a new example each of the n binary classifiers are evaluated, thus a binary string of length n will be produced. Then the resulting string is compared to each codeword (associated with each class), the example is assigned to the class associated with the closest codeword. "Hamming distance" is the most widely used distance measure for this purpose [3]. For simplicity, this method is always represented by a matrix M where each row of M represents a specific class and each column is used to train a binary classifier. The main challenge is to design a good code matrix; a general idea is to have large row and column separation. ECOC has been successfully applied to a several application domains, due to its ability to correct errors generated by the individual base classifiers [33]. In addition ECOC has often been found to be able to outperform the direct use of single multi-class learning algorithms [28, 90]. However, it has been reported that the usage of the simple OVO (AVA) approach produces comparable, or in some cases superior, classification results to ECOC [56].

Another attempt to improve classification accuracy, with respect to using a set of binary classifiers to solve the multi-class classification problem, was the combination of OVO and OVA [40], so called One and All (OAA). The idea behind OAA is to discard the incorrect votes produced using OVO and improve the accuracy of OVA. More specifically, $N(N+1)/2$ binary classifiers are trained, where $N(N-1)/2$ classifiers use OVO and the remaining N classifiers use OVA. For classifying a new unseen example, the example is first classified using the OVA framework and the two classes associated with the highest values are identified. The OVO classifier, corresponding to the identified classes, is then used to classify the example so as to arrive at a final classification decision. Thus the number of classifiers that are required to label a new example is $N+1$. With respect to effectiveness a small improvement has been reported in comparison with OVO and OVA [40]. From the foregoing, OVO (AVA) can be considered as the "state-of-the-art" with respect to the usage of a set of binary classifiers to address the multi-class classification problem.

2.4 Using Ensemble Classifiers to Solve Multi-class Classification Problems

This section provides a review of "Ensemble" methods for solving the multi-class classification problem. An ensemble model is a composite model comprised of a number

[3] Hamming distance refers to counting the number of different bits.

of learners (classifiers), called *base learners* or *weak learners*, that are used together to obtain a better classification performance than can be obtained from using a single "stand alone" model. Classification algorithms such as: decision tree, Naive Bayes, CARM, and neural network can be utilised to generate the base classifiers. If the base learners in an ensemble model are all comprised of the same classification algorithm the ensemble model is referred to as a *homogeneous learner*, while when different classification algorithms are used the ensemble model is referred to as a *heterogeneous learner* [109]. In general, most ensemble methods are categorised as homogeneous learners [109]. Many researchers [47, 49, 57, 78] have demonstrated that generating a "good" ensemble requires base classifiers that tend to make errors on different groups of examples.

Much research work has been directed, by numerous researchers, at ensemble classification due to the potential benefits of the method with respect to classification effectiveness. The history of ensemble methods goes back to 1977 when the idea of an ensemble, made up of two linear regression models, was reported in [97]. More recently Luo and Liu [66] reported that work, using ensembles of neural networks, conducted by Hansen and Salamon [47] was the most significant in the context of better performance and reduced generalisation error[4]. Many researchers have demonstrated that using multiple classifiers reduces the generalisation error [8, 29, 77, 85]. In addition, theoretical evidence that bias-error can be reduced by using ensembles of classifiers was presented in [7]. A novel multi-strategy (hybrid) ensemble, that combined a number of ensemble approaches, reported in [101], noted that ensembles of ensembles were more accurate than their component ensembles.

Although many researchers have demonstrated that ensembles often outperform their "base classifiers" when used on their own [27, 44, 79, 109], few have provided a reasonable answer to the question "why are ensembles superior to stand-alone classifiers?". A suggested answer was provided in [79] that related the better performance of ensembles over single classifiers to the use of all available classification information. A more comprehensive answer was provided by Dietterich in [27],who considered the answer in terms of the following three headings: (i) statistical, (ii) computational and (iii) representational. More specifically:

1. **Statistical reason**. The nature of the data is such that it is often not possible to choose a particular classification model; there are often many different competing classification models that provide the same accuracy on the dataset. Consequently, combining these classifiers produces an average result that is better than that of the individual classifiers. This will avoid choosing the wrong classifier and circumvent the unrelated errors of individual classifiers.

2. **Computational reason**. Using ensembles avoids fruitless, and computationally expensive, searches for the "best" classifier.

[4] Generalization: "The most central concept in machine learning, which characterises how well the result learned from a given training dataset can be applied to unseen new data" [109].

3. **Representational reason**. It is assumed that a given learning algorithm is looking for a "best" hypothesis within the hypothesis space, in most machine learning applications the hypothesis space might not contain the true target function, however adopting an ensemble approach can produce a good approximation.

With respect to the consensus that the ensemble concept is a general methodology for improving the accuracy of "stand-alone" classification algorithms; the ensemble approach is applicable and can be employed in all areas where classification techniques can be applied. Examples of application domains where ensemble have been used include: text categorisation [73], bioinformatics [106] (due to their ability to deal with high-dimensionality and complex data structures), manufacturing [69], e-learning evaluation system [54], and medical diagnosis [93].

According to Rokach [89] four main factors can be used to characterise the various ensemble methods:

1. **Inter-classifier relationship**. This refers to the relationships between classifiers forming the ensemble and how these classifiers affect each other. Two main types of ensemble can be differentiated: concurrent (parallel) and sequential (cascading). The hierarchical ensemble, which is a much more recent approach, can be considered as a special case of a sequential ensemble. Most proposed ensemble models fall into the concurrent category. In a concurrent ensemble the classifiers are independent and their results are combined together using some combination scheme (see factor 2 below). In a sequential ensemble the classifiers are arranged sequentially (or hierarchically). More details concerning concurrent, sequential, and hierarchical ensembles are provided in the following sub-sections.

2. **Adopted Combination scheme**. Regardless of how an ensemble system might be configured, an important issue is how results are combined to produce a final classification. The simplest approach is to use some kind of voting system [8]. Voting algorithms can be divided into two types: those that adaptively change the distribution of the training set based on the performance of previous classifiers (as in boosting methods) and those that do not (as in Bagging). Averaging is another scheme to combine the results of several classifiers, which is suitable for use with classifiers that generate (say) confidence or probability values. A more complicated combination method can be adopted that utilises the concept of a "meta learner" such as stacking [105]. Stacking is usually used to combine models of different types, however it is not widely used.

3. **Ensemble size**. This refer to the number of classifiers forming the ensemble. A number of issues should be taken into consideration here: (i) accuracy, (ii) computational complexity and (iii) the number of available processors. Some researchers have claimed that the usage of large numbers of classifiers improves classification accuracy [47], however this is clearly not true with respect to the disjoint partitioning methods, where if the subset sizes are too small, insufficient information will

be available for learning effective classifiers with which to populate the ensemble [89].

4. **Diversity**. The concept of the diversity of an ensemble refers to the generation of a set of base classifiers that are as diverse as possible so that they will produce uncorrelated errors; it is suggested that consequently a better overall effectiveness (classification accuracy) can be obtained [51]. The simplest way to obtain a diversified ensemble is to use different representations of the training data. In other words manipulating the training examples, as in bagging where each classifier is learned using a different subset of the original training data. Manipulating the attribute set is another way of obtaining diversity, however it is not commonly used. The idea is to assign a different attribute set to each classifier [89].

Before continuing with the discussion on the usage of ensemble classifiers to solve the multi-class classification problem a number of open issues associated with the ensemble methodology should first be considered, these can be summarised as follows:

1. **The best way to construct ensemble of classifiers.** It is generally acknowledged that there is no "best" ensemble, the reason for this is simply because there is no "best" classification algorithm. However, some researchers have recommended ways of constructing ensembles for specific situations, for example one study recommend not using sequential ensemble when the data set is highly noisy [85].

2. **No comprehensive comparison available in the literature.** The available studies vary regarding to the used: (i) ensemble approaches, (ii) evaluation data sets and (iii) evaluation criteria.

3. **Computational cost.** It is clear that combining a set of classifiers is computationally more expensive than using single "stand-alone" classifier. However, the promising benefit, obtaining accurate classification, generally considered to be worthwhile. In order to address the issue of complexity associated with ensemble systems two options have been suggested: (i) the usage of parallel processing, especially for concurrent ensembles as suggested by Breiman [12] and (ii) the elimination of similar representations from ensembles of classifiers, in other words *pruning*, as suggested by Dietterich [26].

4. **Difficulty in understanding the final classification decision.** For example, and as noted by Dietterich [26], it is easy to understand the classification result of a single decision tree. However, it is difficult to understand a final classification result of an ensemble comprised of two hundreds decision trees.

The rest of this section is divided into four parts. Part 1 provides a general overview of concurrent ensembles and presents the most popular concurrent ensemble approaches.

Then Part 2 goes on to consider sequential ensembles and provides a review of the most well known sequential ensemble approaches. Part 3 then provides a detailed survey of the domain of binary tree based hierarchical ensemble classification. Followed, in Part 4, by a discussion of DAG based hierarchical ensemble classification. The reason for this division is that the work on hierarchical ensemble classification presented later in this thesis can also be divided into binary tree and DAG based approaches.

Part 1: Concurrent Ensemble Methods

Using the concurrent ensemble methodology, the original data set is divided into several partitions, either disjoint (mutually exclusive) or overlapping. Each partition is used to learn a classifier. More specifically, several classifiers are trained concurrently. When classifying a new unseen example the final classification result will be some combination of individual classification results. The main goal of the parallel ensemble methodology is to improve the classification performance, both in terms of effectiveness and efficiency [70]. The latter, can be realised when utilising some form of parallel or distributed processing.

The simplest and the first proposed parallel ensemble method is the "Bagging" method (the name was obtained from the phrase "Bootstrap Aggregation") [67]. In bagging what is termed "sampling with replacement" is used [12]. What this means is that each classifier is trained using a different random sample of the training data set (note that the same example may be sampled more than once). More specifically, the original training data set is divided into N samples of the same size as the original training data set. Thus the sampling process might result in: (i) the appearance of some examples more than once within the same sample and/or (ii) the non-appearance of some examples in any sample. With respect to classifying new unseen example a simple voting process is usually adopted. The following advantages can be identified for bagging ensembles: (i) implementation simplicity and (ii) improving classification performance (efficiency and effectiveness). A well-known bagging algorithm is the "Random Forest" algorithm [13], which comprised of a collection of decision trees. The reasons behind the popularity of the random forest algorithm are: (i) relatively low computational cost and (ii) ability to achieve excellent classification performance compared with many other classification methods [16].

There are some variations of bagging, often referred to as "Bagging-Like-Strategies", which handle smaller sized partitions of the training data. As in the case of standard bagging, the original training data set is divided into N subsets of the same size, and each is used to train an individual classifier. The combination of the individual classifiers produces a composite classifier. From the literature we can identify four different methods for conducting "Bagging-like" partitioning [67]:

1. **Disjoint partitions**. In which an example can be found only in one partition and only once within that partition.

2. **Small bags**. In which an example can be found in several partitions and/or several times within a partition.

3. **No replication**. In which an example can be found in one partition, several partitions, or no partition; however, an example can only be found once within the same partition.

4. **Disjoint bags**. In which an example can be found several times within the same partition, but cannot be found in other partitions.

A more intelligent partitioning technique based on clustering was proposed by Chawla et al. [19] where clustering was used to create "meaningful" partitions of the original data set. More specifically the algorithm identified a set of clusters (partitions) from the original data set, a classifier was then trained using each partition. The suggested technique was compared with a simple data partitioning techniques that split the data set into random disjoint parts (random splitting). The reported results showed that the clustering based method outperformed the random partitioning method and C4.5 decision tree classification.

The significance of the partitioning methods discussed in this sub-section, with respect to the work presented later in this thesis, is that the proposed hierarchical partitioning methods also feature partitioning. However, the partitioning is directed at grouping classes rather than examples.

Part 2: Sequential Ensemble Methods

Using the sequential ensemble approach, unlike in the case of the concurrent approach, there is interaction between the different classifiers (the outcome of one feeds into the next). The main conjectured advantage is that the knowledge produced in a previous iteration can be utilised to enhance the training process in the next iterations. In this section a number of well documented sequential ensemble methods are considered, namely: (i) boosting, (ii) windowing and (iii) stacking.

One well studied form of sequential ensemble classification is known as "Boosting", where a sequence of weak classifiers is "chained" together to produce a single composite strong classifier in order to achieve a higher combined accuracy than that which would have been obtained if the weak classifiers were used independently. A well-known boosting algorithm is Adaboost [35]. The central idea of Adaboost is to assign a weight to each training example. Initially, all examples weights are equal, however in each iteration the weights are adjusted to reflect the effectiveness of the corresponding classifiers. More specifically, the weights of the mis-classified examples are increased, and the weight of the correctly classified examples decreased. The central goal is to enforce the classifier to focus on the "difficult" examples, so the resulting classifiers "complement" one another [89]. The final composite classifier combines the base classifiers by voting, but each classifier's vote is based on its accuracy. The following advantages can be identified for

AdaBoost: (i) implementation simplicity, (ii) flexibility, (any classification algorithm can be used to produce the base classifiers) [35], (iii) ability to identify outliers (examples with the highest weight are always considered to be outliers) [35], (iv) the generated composite classifier (ensemble) has fewer classification errors than the base classifiers [89] and (v) versatility (the ensemble can be applied to a wide variety of applcations). On the other hand it has been suggested that, a large number of iterations may generate a very sophisticated classifier that tends to be less accurate than a single classifier [85]. Regardless of this disadvantage, "Boosting" is one of the most widely used sequential ensemble methods.

Windowing is another example of a sequential ensemble method. Windowing was proposed to enable the ID3 decision tree classifier to address classification problems that requires very large memory capacity [84]. Windowing commences by selecting a random subset of the training data set, a "window". The subset is then used to train a classifier. The next step is an evaluation step, in which the generated classifier is evaluated using the remaining training examples. If the obtained accuracy is "not sufficient", all the misclassified examples are removed from the evaluation set and added to the window for the next iteration. This procedure is repeated until "sufficient accuracy" is obtained. Note that the last trained classifier is considered to be the final classifier. Windowing did not gain much attention in the machine learning field, in comparison to "Boosting" or even other ensemble methods, due to: (i) the fast development of computer hardware which has made more memory available and (ii) a dedicated empirical study, which applied windowing using ID3 with respect to several domains, that found that windowing did not enhance classification efficiency [104].

"Stacking" [105] can be considered as another form of sequential ensemble approach. More specifically it is: (i) a combination of the sequential and concurrent ensemble approaches (provided parallel processing is applied in the first stage) and (ii) a way of combining results from ensemble classifiers. The central idea of stacking is to generate a "meta-dataset", using an ensemble of classifiers referred to as the "first-level" that serves as input to a "second-level" classifier. More specifically, given a training data set of N examples, the training procedure is as follows:

1. Use $N - 1$ examples to train the first level classifiers.

2. Classify the leave-out example using all the first-level classifiers. The prediction results are then used to form a meta-example for the corresponding leave-out example. More specifically, the class labels, resulting from the first-level classifier, will form the attributes of the meta-example. Note here that the class label of the example will be maintained as in the original training data set.

3. Repeat the previous step for all examples in the training data set. Consequently, a corresponding "meta-example" will be produced for each training example.

4. Use the resulting "meta-dataset" to learn the second-level classifier.

5. Train the first level classifiers using all examples in the training data set (N examples instead of $N-1$). The objective here is to use all available information to train the first-level classifiers.

For classifying a new example, the first-level classifiers classify the example, then the classification results (class labels) are passed onto the second-level classifier to produce the final prediction.

Part 3: Binary Tree Based Hierarchical Ensemble Methods

As noted earlier in this chapter, a more recent approach to solving the multi-class classification problem involves the creation of hierarchical ensemble classifiers [4, 20, 58, 60, 68, 100]. A common structure adopted for hierarchical classification is a binary tree constructed in either a bottom-up or top-down manner [11, 58]. In a binary tree hierarchical classification model, as the name suggests, the classifiers are arranged in a binary tree formation. As noted above the desired binary tree structure can be constructed in either a bottom-up or a top-down manner; however, top down construction is the most widely used because it tends to produces a more balanced structure and because it is easier to implement [4]. Using the top down approach the process is as follows: starting at the root of the tree, a grouping technique is used to segment the examples into two clusters, each cluster is labelled with a group-class label. Then, a classifier is trained to classify examples using the two group-classes. The process continues recursively until classifiers are arrived at that can assign single class labels to individual examples. The bottom-up model comprises a merging process similar to agglomerative hierarchical clustering. On each iteration the two most similar nodes are merged to form a node describing a new meta-class [11]. For classifying a new example a "path" is followed from the root, according to the classification at each hierarchy node, until a leaf node associated with a single class label is reached.

Before continuing with presenting the previous work with respect to the binary tree hierarchical classification model, an example binary tree hierarchy will first be presented so that the reader can understand the general principles of the model. An example binary tree is given in Figure 2.1. To create such a binary tree the hierarchy a classifier needs to be generated for each node in the hierarchy using an appropriately configured training set. The process is illustrated in Figure 2.1 where the root classifier classifies the data set into two sets of class labels $\{a, b, c, d\}$ and $\{e, f, g\}$, the level two classifiers are then directed at further subsets and so on. The sets of class labels (the label groupings) are identified by repeatedly dividing the data using a data distribution technique, typically founded on ideas concerned with clustering and splitting techniques. Note that the nature of the classifiers held at each node can be of any form.

Having established the general concept of the binary tree hierarchical classification model the previous work regarding such trees will be discussed. The work on binary tree hierarchical classification can be differentiated depending on: (i) the classifiers used at nodes and (ii) the adopted technique to distribute class labels between nodes

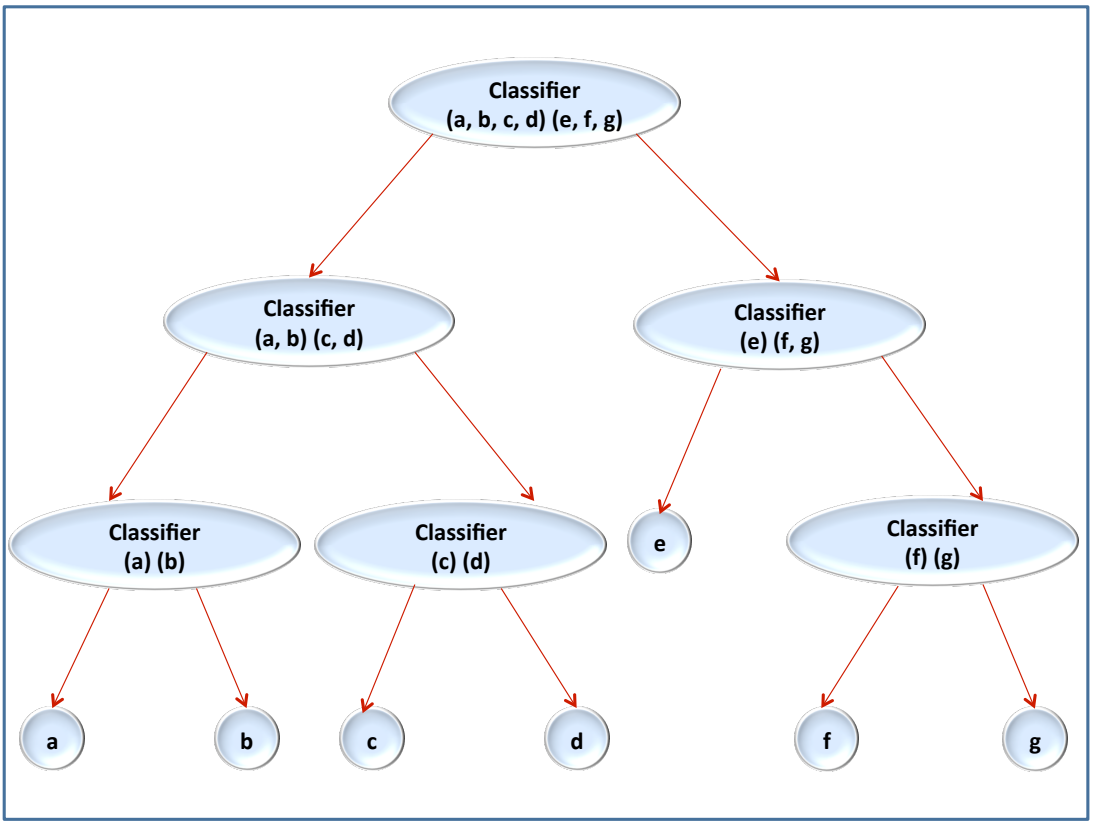


FIGURE 2.1: Binary Tree Hierarchy example.

within the hierarchy. With respect to the adopted technique for distributing class labels between nodes within a binary tree these can be categorised according to whether they permit overlapping between classes at nodes or not. The work presented in this thesis considered both categories. With respect to the classification strategy, the strategy for classifying a new unseen example, the previous work only considered following a single path within the binary tree hierarchy. While the work presented in this thesis considers following single and multiple paths, this will be discussed in detail in Chapter 3. In addition, the previous work on Binary Tree hierarchical classification only considered using stand-alone classifiers at each hierarchy node, while the work presented in this thesis also proposes using ensembles of classifiers at each node. From the literature a number of approaches to binary tree based hierarchical ensemble classification can be identified: (i) Binary Hierarchical Classifier (BHC), (ii) Hierarchical Support Vector Machines (HSVM), (iii) Divide-By-2 (DB2), (iv) Half-Against-Half (HAH), (v) Support Vector Machines Binary Decision Tree (SVM-BDT) and (vi) Binary Classification Tree with Observation based clustering (BCT-OB). Table 2.1 provides a general overview of these approaches, while the remainder of this section presents a detailed explanation for each one. Note that the reason for the inclusion and explanation of these approaches to binary tree based ensemble classification, presented in this previous work chapter, is to differentiate the work presented in this thesis from the previous work with respect to binary tree hierarchical classification.

TABLE 2.1: Binary Tree hierarchical classification approaches

Name	**Classifiers at Nodes**	**Data Distribution Technique**	**Overlapping/Non-overlapping**
BHC	Stand-alone Naive Bayes	Bayes rule	Non-overlapping
HSVM	Stand-alone SVM	Max-Cut hierarchical decomposition	Non-overlapping
DB2	Stand-alone SVM	K-means, Spherical shells and Balanced subset	Non-overlapping
HAH	Stand-alone SVM	Hierarchical clustering	Non-overlapping
SVM-BDT	Stand-alone SVM	Distance between classes gravity centers	Non-overlapping
BCT-OB	Stand-alone SVM	K-means	Overlapping (but restricted)

Commencing with the work suggested by Kumar et al. [58], the Binary Hierarchical Classifier (BHC) approach. The BHC is a binary tree hierarchical classification approach where the ensemble is generated in a top-down manner. Given a data set with N classes, the resulting binary tree has N leaf nodes, one for each class, and $N-1$ internal nodes. Kumar et al. implemented the BHC approach using a Bayesian classifier at each tree node; Bayes rule was utilised to partition the classes at each tree node into two disjoint subsets, referred to as "meta-classes". More specifically, each class was treated as an object, a class is assigned to one of the two meta-classes based on posterior probability that the class belonged to a specific meta-class. The assignments of the classes are updated several times over multiple iterations. It is interesting to note here that, in addition to the Bayesian classifier held at each internal node, a linear feature extractor was also used. The objective of the latter was to extract features from remotely sensed hyper-spectral data[5] that was used to evaluate the proposed BHC model. Classes partitioning and feature extraction were conducted simultaneously (coupled within the same algorithm). According to the reported evaluation, BHC operated in a significantly better manner than approaches based on other feature extraction and problem decomposition techniques. Further empirical studies reported that BHC: (i) was applicable to the multi-class classification problems and (ii) that the obtained results were comparable (both in terms of efficiency and effectiveness) to ECOC [86] (as described in Section 2.3). The main motivation for this work was to address the problem of high dimensional

[5] 180 dimensions and 12 classes

data not particularly the multi-class classification problem. Consequently, the data distribution technique, which was combined with a feature extraction procedure, does not permit overlapping between class labels to reduce the complexity of the problem. While the work presented in this thesis considers both overlapping and non-overlapping set of class labels using different data distribution techniques.

Hierarchical Support Vector Machines (HSVM) [20] are a form of binary tree hierarchical classification. The HSVM approache comprises: (i) a max-cut hierarchical decomposition and (ii) SVM classification. More specifically, the input data is considered as an undirected graph where nodes represent classes and edges represent the average "Kullback-Leibler" distance between the density function of the two classes at the end nodes. A Max-Cut hierarchical decomposition method is applied to split the classes (graph nodes) into two partitions. This is done by identifying the maximum total distance between two class partitions (maximum total pairwise distance measure). The max cut procedure is applied recursively, thus a binary hierarchical decomposition is achieved. The reported evaluation results showed that the HSVM approach achieves high classification accuracy when sample sizes are small in terms of number of attributes and the number of classes is large. Again, as the case of BHC, this approach was applied to hyperspectral data and no overlapping between classes was permitted.

Vural and Dy [100] introduced a binary tree based hierarchical classification called Divide-By-2 (DB2). The main motivation for this work was to extend the use of support vector machines (SVM) to address multi-class problems. DB2 utilised three different techniques to divide the data into two subsets at each hierarchical level:

1. **K-means.** A technique whereby each class is represented with its corresponding mean[6], the k-means algorithm is used to group the class means.

2. **Spherical shells.** This is a technique whereby a threshold is used to group the classes. Again, each class is represented by a class mean. The threshold is calculated as the mean of classes means. Then the procedure is as follows. The class associated with a class mean smaller than the threshold is considered to be the negative class, while the class associated with a class mean greater than the threshold is considered to be positive class.

3. **Balanced subset.** This is a technique whereby the data is divided into two subsets such that the difference in the number of examples in each subset is minimised.

No overlapping of classes between subsets was permitted, thus the number of the classifiers to be trained was $N - 1$ where N is the number of class labels in a given data set. For the evaluation purpose, DB2 was compared with: (i) OVO, (ii) OVA and (iii) SVM Directed Acyclic Graph (SVM DAG). The reported evaluation indicated that DB2 is always faster than OVO and OVA in terms of classification time, and it is faster

[6]The class mean/class center is a "prototype" example derived from the means of the attribute values for the examples/examples that belong to that class.

than DAGSVM when the data set is unbalanced. With respect to effectiveness, DB2 produced a comparable accuracy results to the alternative methods considered.

Half-Against-Half (HAH) is another binary tree hierarchical classification approach suggested by Lei and Govindaraju [60]. HAH use an SVM classifier at each tree node and hierarchical clustering to distribute the class labels into two subsets. As in the case of the earlier work conducted by Vural and Dy [100]; the classes are divided into two disjoint (no overlapping) partitions based on a distance measure. The distance between two classes is defined as the mean of the distance between the training examples for the two classes. Lei and Govindaraju provided a theoretical study, and experimental results, indicating that: (i) HAH is more efficient than OVA, DAG SVM, and OVO in terms of classification and generation speed; and (ii) with respect to effectiveness, HAH is comparable to OVO, SVM DAG and OVA, in terms of classification accuracy. Again, as in the case of the previous approaches, no overlapping between the class groups is permitted.

Another approach, similar to DB2, that utilised SVM and distance measures to generate a binary tree hierarchical classifier is the Support Vector Machines Binary Decision Tree (SVM-BDT) approach reported in [68]. Using the SVM-BDT approach the classes are divided into two disjoint groups by calculating N gravity centres for the N different classes. Then, the two classes with the highest distance from each other are assigned to each of the two groups. Next, as in the case of some clustering procedures, the remaining classes are assigned to groups using a distance measure. The centre of each group is updated on each occasion. The process continues until all classes are assigned to one of the two possible groups. The grouping procedure is repeated at each tree node until only one class is left in each group, this then represents a leaf node in the tree. Because no overlapping of classes across groups is permitted, only $N-1$ classifiers need to be trained. SVM-BDT gains advantages from: (i) the efficient computation of the decision tree architecture and (ii) the high classification accuracy of SVMs. For evaluation purpose SVM BDT was compared with: (i) SVM based approaches (OVO, OVA, and SVM DAG), (ii) Ensemble of trees (Random Forest) and (iii) neural network based approaches. The reported evaluation indicated a comparable or better accuracy, and improved generation and classification times for the proposed SVM-BDT approach. Again no overlapping between classes was permitted.

A more recent approach to binary tree hierarchical classification is the Binary Classification Tree with Observation based clustering (BCT-OB) approach; a novel approach to tree splitting [4] whereby, unlike the previous approaches, classes are partitioned by performing clustering on examples instead of class means (centers). Thus, classes can appear in both clusters (overlapping) and as a result it will be considered in both subtrees. K-means and SVM are the two algorithms used with respect to the BCT-OB approach. The data distribution technique, at each tree node, is conducted as follows:

1. K-means clustering is applied to the examples, with $K = 2$, to give two clusters.

2. A data cleaning process is performed. More specifically, the proportion of each class in each cluster is calculated by dividing the number of examples of class x in cluster i with the number of examples of x. Examples belonging to a specific class in a specific cluster where the proportion value is less than a predefined threshold will be eliminated from the cluster.

The advantages of the BCT-OB approach can be summarised as follows: (i) allowing some classes to be addressed in different sub-trees offers an opportunity to detect any sub-patterns in a class and (ii) the cleaning process prevents redundant learning (from small sub-groups) and also serves to reduce the computation time. On the other hand the limitations of the BCT-OB approach are as follows: (i) the performance is highly affected by the adopted clustering algorithm (K-means or otherwise) and (ii) the threshold value affects both tree construction and classification performance, a larger tree will be produced if a low threshold is used than when a high threshold is used. Also, classification accuracy will be decreased if the threshold is too high due to information loss. In comparison with other binary classification tree algorithms the reported experimental results showed that BCT-OB performs comparably. To the best knowledge of the author this work is the only previous work that considers overlapping between class groups, as also considered later in this thesis. The difference between the work presented in this thesis and the BCT-OB algorithm is that, in the work presented in this thesis, no threshold was used to eliminate examples from a class group (cluster). The reasons for this were: (i) the threshold value can highly affect classification accuracy as reported in [4] and (ii) the threshold value results in eliminating some examples from a class group, thus the number of examples available for the training process will decrease (information loss).

The foregoing binary tree hierarchal ensemble classification approaches can be considered to be significant with respect to the work presented in this thesis because a groups of class labels are addressed at each binary tree node (except leaf nodes). A less significant work, with respect to the work presented in this thesis, that made use of binary tree structures to combine the results of a set of binary classifiers [72]. This can be viewed as a special case of using a set of binary classifiers to handle the multi-class classification problem. More specifically, given a data set with N classes, all pairwise classes are identified and handled by using a set of binary classifiers. However, instead of applying voting to combine the predictions, as in OVO, the binary tree structure (Directed Binary Tree (DBT)) was utilised for this purpose. An example of DBT is presented in Figure 2.2. For classifying a new example, starting from the root, one class is eliminated at each level until a leaf node is arrived at. Then leaf class will be assigned to the example. Some researchers have tried to find strategies to determine the optimal order of the classifiers within the binary tree so as to obtain a better overall classification accuracy. The proposed ordering strategies have been mainly focused on some features of SVM classifiers such as margin size or number of support vectors (classifier-dependent attributes) [72]. Note that the reasons behind the inclusion and explanation of DBT

approach here are: (i) to differentiate it from the work presented later in this thesis, where both used binary tree structure to solve the multi-class classification problem and (ii) DBT approach was proposed to solve the multi-class classification problem which is the main motivation of this thesis.

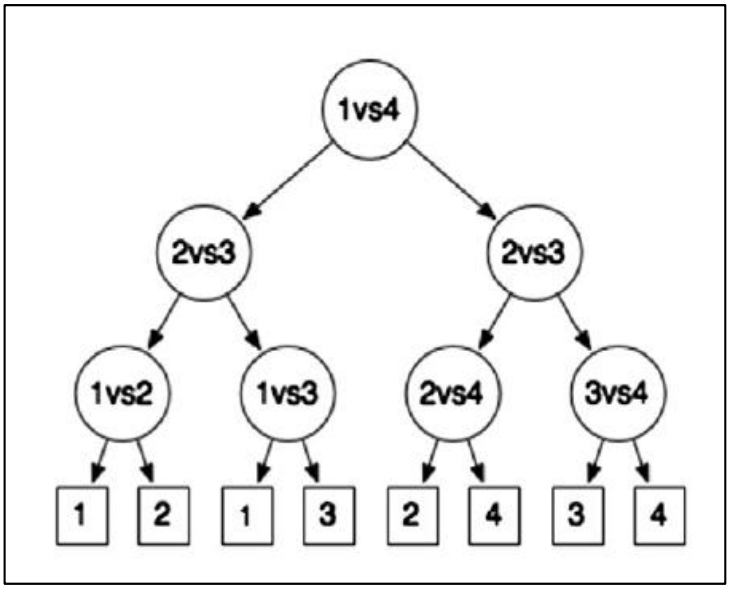


FIGURE 2.2: Directed Binary Tree (DBT) example [72]

As noted earlier in this section, the previous work, with respect to binary tree hierarchical classification model, can be differentiated according to the adopted classification algorithm, and the manner in which the classes are distributed between nodes within the hierarchy. From this previous work, the following can be noted:

1. SVM is most frequently used as the base classification algorithm. There is little reported work in using other forms of classification algorithms such as decision trees, neural networks, or Naive Bayes algorithms. Much of the SVM based work on binary tree ensembles is directed at allowing the application of SVMs (essentially a binary classification algorithm) to the multi-class problem.

2. A straight forward splitting techniques is the most frequently used methods for dividing the classes between nodes (without overlapping between the class groups). Thus the number of required classifiers can be determined previously ($NumOfClassifiers = N - 1$).

3. To the best knowledge of the author, the splitting or clustering techniques reported on in the literature, for dividing classes between nodes, are always applied to the classes centres (means) not directly to the examples themselves. Only one recent work applied clustering algorithms directly on examples [4].

4. From the literature the operation of the resulting hierarchical ensembles are always compared with OVO and OVA instead of comparing with other ensemble methods

such as Bagging or Boosting, the author has only found one published paper which considered comparison with other forms of tree ensembles (bagging and random forest) [68].

5. No work has been found by the author that addresses the most significant drawback of the hierarchical model, which is that if an example is mis-classified early on in the process it will continue to be mis-classified throughout the process.

6. Relating to the previous point, only single-paths are followed within the hierarchy.

Part 4: DAG Based Hierarchical Ensemble Classification Methods

This section presents previous work that has utilised DAG structures to solve the multi-class classification problem. As will be seen, this previous work is significantly different than the work presented in this thesis, with respect to utilisation of DAG structures for hierarchical ensemble classification. More specifically the previous work on DAG hierarchical ensemble classification has focused on utilising a DAG structure to combine the prediction results obtained from a set of binary classifiers, while with respect to the work presented in this thesis, groups of class labels are considered at each DAG node not two classes (binary classification), this will become more apparent later in this thesis. The work presented in this section can be considered to be a special case of using a set of binary classifiers to solve the multi-class classification problem presented in Section 2.3. The reason behind the inclusion and explanation of this work here is to differentiate it from the work presented later in this thesis.

This section presents three examples of previous work that has utilised DAG structure to solve the multi-class classification problem: (i) Decision Directed Acyclic Graph (DDAG), (ii) Adaptive Directed Acyclic Graph (ADAG) and (iii) Reordering Adaptive Directed Acyclic Graph (RADAG).

Platt et. al. [80] were the first to suggest using a (rooted) Directed Acyclic Graph (DAG) for hierarchical ensemble classification in 2000. More specifically, the rooted DAG structure was utilised to arrange several binary classifiers into a single classifier called a DDAG (Decision Directed Acyclic Graph) [80]. In DDAG the nodes are organised in a triangular form, starting with single node at the top (root), two nodes at the next level, and so on, till arriving at the last level (the leaf level) where the individual class labels are represented (one per node). Each DAG node holds a SVM classifier. The algorithm used to generate the DDAG, the SVMDAG algorithm, generates $N(N-1)/2$ classifiers for N class labels, as in the case of OVO, where every pairwise classes is considered. Figure 2.3 presents a DDAG example. To classify a new example, starting at the root of the DAG, the first binary classifier classifies the given example and consequently we proceed down the left or right branch depending on this classification result. The process continues until a leaf node is reached that holds the predicted class. More specifically the classes are ordered arbitrarily in a list. Reordering the classes was found not to result in any significant change in accuracy. Classification process adopted in [80] is essentially

an *elimination process*; one class will be eliminated from the list at each node. At the root node the list contains all the classes, the root classifier classifies the new example either to the first or the last class in the list. If the predicted class is one of the two classes, the other is eliminated, the process continues with respect to the first and the last classes in the list, until only one class remains, which is the predicted class for the new example and the SVMDAG algorithm terminates.

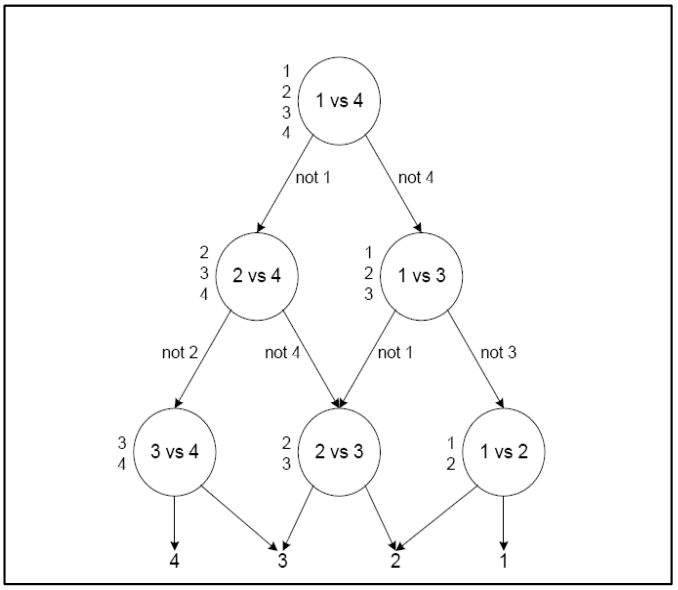


FIGURE 2.3: Decision Directed Acyclic Graph (DDAG) example [80]

An issue with the DDAG, as reported in [55], is that the number of nodes that needed to be evaluated in order to obtain a final resulting class label is "unnecessarily" large, and affects the classification accuracy. More specifically, the number of times that the correct class should be evaluated against the rest is $N-1$ (the depth of the DDAG) where N is the number of the classes in a given dataset. Consequently, the probability of cumulative error increases.

To address this problem Kijsirikul et. al. suggested the Adaptive Directed Acyclic Graph (ADAG) which modified the DDAG to obtain better classification accuracy by reducing the number of classifiers that needed to be evaluated before arriving at a final classification decision. Using an ADAG the nodes are organised in a reverse triangle. Given N class labels, the top level will include $N/2$ nodes (rounded up), $N/2^2$ nodes at the second level, and so on, till arriving at one node at the last level. Thus, ADAG generates $N(N-1)/2$ binary classifiers, as in the case of SVMDAG, arranged in $N-1$ internal nodes. Figure 2.4 presents an ADAG example. For classifying a new example the process commences at the top level by evaluating all nodes. As a result a preferred class will be passed on to the next level, where the number of possible classes is reduced by half. A node in this next level is then selected according to the passed class. The process continues till the final level is arrived at where only one node exist, evaluating this node will result in the class label for the new example. Reported theoretical and

experimental results have shown that the ADAG structure gives a better classification accuracy than the DDAG structure, especially when the number of class labels is large.

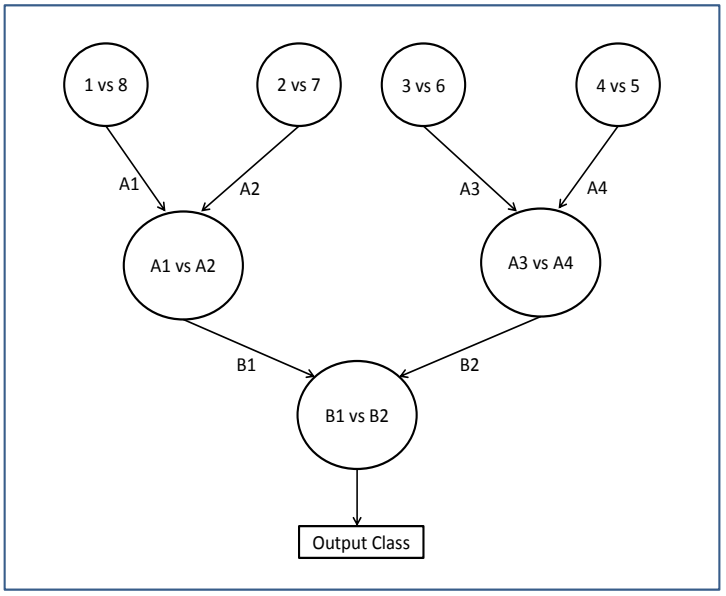


FIGURE 2.4: Adaptive Directed Acyclic Graph (ADAG) example [55]

Although it was reported that the ADAG approach produced a better classification accuracy than DDAG by reducing number of the classifiers that the correct class should be evaluated against (only $\log_2 N$, while $N-1$ for DDAG), the sequence in which the classifiers at nodes were invoked still affecting the classification accuracy. In order to resolve this node sequence dependency, the Reordering Adaptive Directed Acyclic Graph (RADAG) approach was proposed by Patoomsiri et. al. [92]. Here what was referred to as generalisation error, which is the actual performance of the classifier when evaluated on unseen data[7], was utilised to select the optimal classifiers. The RADAG approach is similar to the ADAG approach except: (i) the initialisation of the classifier at the first level and (ii) the order of the classes in the lower levels. The generalisation errors of all possible pairs of classes are calculated at the beginning of the algorithm, using k-fold-cross-validation. At each level the classifiers that have the smallest sum of generalisation errors will be used in the evaluation process. Reported experimental results demonstrated that the RADAG approach generates higher classification accuracies than ADAG. Generalisation error has also employed in DDAG, to select the most suitable classifiers in the evaluation stage. Two models to enhance DDAG were suggested based on the use of generalisation error, Strong Elimination (SE) of Classifiers, and Weak Elimination (WE) of Classifiers [92]. Classifiers associated with the minimum generalisation error are evaluated first. The distinction between SE and WE is that in SE, at each evaluation level, the classifiers related to the eliminated class are ignored (the same procedure as in the case of the original DDAG approach). While when using WE a classifier will be ignored only if its two associated classes are eliminated, the reason

[7] Generalisation error = number of mis-classified examples / number of examples.

behind this is that classifiers associated with high generalisation ability can be helpful when eliminating the remaining candidate classes. Experimental results demonstrated that both SE and WE produce a higher accuracy than the DDAG approach; best results were produced using the WE method.

2.5 Comparison with Previous Work on Binary Tree Based Hierarchical Ensemble Methods

From Section 2.4 (Part 3) it was noted that the previously proposed binary tree based hierarchical ensemble classification methods can be categorised into two main categories with respect to the adopted technique to distribute class labels between nodes within the binary tree: (i) overlapping between classes at nodes and (ii) no-overlapping between classes at nodes. Note here that only one reference was found by the author that considered overlapping between classes [4]. The work presented in this thesis considered both categories. In addition, with respect to the previous work, the adopted techniques to group class labels was focused on grouping similar classes together early on in the process so that entire branches ended up dealing with very similar classes (as in the case of the proposed clustering techniques presented in this thesis). Ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built at each leaf node. This issue has not been reported previously. Because of this issue a simple data splitting technique was proposed in this thesis. Moreover, the previous work only considered following single paths within the binary tree hierarchy, while the work presented in this thesis considered following single and multiple paths within the binary tree hierarchical classification model.

2.6 Clustering

This section provides a generic overview of the clustering concept. The significance is that clustering techniques are, in some cases, used by the hierarchical ensemble approaches presented later in this thesis to distribute classes between nodes. Clustering is an unsupervised learning process that aims to partition a set of examples into groups (clusters), so that examples that belong to a single cluster are in some sense similar to each other and dissimilar to examples in other clusters [31]. Although it is difficult to categorise clustering techniques according to the nature of their operation, because many clustering techniques share elements of their mode of operation with other techniques [44], four basic categories can be identified:

1. **Partitioning techniques**. Partitioning techniques are considered to be the most widely used [5]; the idea is to produce disjoint clusters (each example belongs to a distinct cluster). Commencing with the initial partitioning, examples are reassigned to clusters till a specific criteria, usually distance-based, is arrived at [70]. K-means is the most commonly used partitioning algorithm.

2. **Hierarchical techniques**. These techniques operate in a hierarchical manner by grouping examples either in a top-down (divisive) or bottom-up (agglomerative) manner [44].

3. **Density-Based techniques**. The goal of these techniques is to discover arbitrarily shaped clusters not necessarily convex shaped clusters (spherical-shaped clusters) [70].

4. **Grid-Based techniques**. These methods, as the name suggests, use a grid data structure (comprised of multi-rectangular cells) to partition the problem space into cells. Examples within a cell are represented by the cell. Clustering is applied to the cells instead of the data. Consequently, low processing time is required, in comparison with other clustering techniques [10].

Among the above clustering techniques, partitioning and hierarchical techniques are of interest with respect to the work described in this thesis because: (i) partitioning algorithms, specifically k-means, are the most well-known and widely used and (ii) hierarchical algorithms, specifically *divisive* hierarchical clustering, fit well with respect to the vision of hierarchical ensemble classification presented in this thesis. A more detailed explanation of k-means and *divisive* hierarchical clustering is thus presented in the remainder of this section.

Commencing with the k-means algorithm, this partitions a data set into k clusters, thus k must be specified previously. Each cluster is associated with a center (also called the cluster centroid, mean or center-point). Examples are assigned to the cluster associated with their closest centroid, distance metrics are typically used for this purpose. The procedure can be summarised as follows:

1. Randomly Select k examples to be the initial clusters centers (means).

2. Assign each remaining example to the closest cluster (formation of the clusters).

3. Update the centre (mean) for each cluster.

4. If the centres (means) change, repeat from 2 (the process terminated when no changes in clusters centers).

With respect to *divisive* (top-down) hierarchical clustering, the examples are recursively partitioned in a top-down hierarchical manner (which can be illustrated using a dendrogram). The process commences with all examples in one cluster, on each successive iteration, a cluster is split into smaller clusters, based on some metric (such as cohesion or dissimilarity), until a "best" cluster configuration is arrived at. The "best" configuration indicates a configuration where either: (i) each example is in its own cluster or (ii) the examples in a cluster are sufficiently similar [44]. In order to measure distance between two clusters a number of measures have been proposed, such as the single-link (also called minimum distance or nearest-neighbour) and complete-link measures [53].

Using the single-link measure the distance between two clusters is the minimum of the distances between all pairs of examples in the two clusters. Using the complete-link measure the distance between two clusters is the maximum of all pairwise distances between examples in the two clusters [53].

2.7 Overview of Statistical Tests

This section provides a generic overview of the statistical tests used to compare different classification models. Several statistical tests are available for the purpose of comparing the operation of classifiers; the question is which one to adopt? A comprehensive theoretical and practical study of the available statistical tests for comparing machine learning algorithms was conducted by Janez Demsar [25]. According to this study, non-parametric statistical tests are recommended for classification algorithm comparison purposes. The reasons behind this recommendation were that these tests do not assume: (i) normal distributions of the samples across a set of problems, and (ii) homogeneity of the variance (random variables have equal variance). More specifically, the Wilcoxon signed rank test was recommended for comparing two classifiers, while the Friedman test, with a corresponding post-hoc test, was recommended for comparing several classifiers; more than two over multiple datasets. A more detailed explanation of the Wilcoxon signed rank test and the Friedman test is thus presented in the remainder of this section.

The Wilcoxon signed ranks test is a non-parametric test to compare two classification algorithms over multiple datasets [103]. Using this test: (i) the differences in effectiveness of the two considered classifiers are calculated, (ii) a ranking is applied according to the calculated differences, and (iii) a comparison of the positive and negative ranks is applied.

The Friedman test is a non-parametric test to compare multiple classifiers over multiple datasets [37]. As in the case of the Wilcoxon test, this test is based on a ranking process. More specifically, each classifier is given a *rank* for each dataset, the average rank is calculated for each classifier, a comparison of the calculated ranks is applied. If, as a result of applying the Friedman test, a significant difference is detected between the considered classifiers[8], a post-hoc test is applied to determine which classifier(s) is (are) significantly different than the others. According to Demsar [25], the usefulness of any post-hoc test will be better if the classifiers are compared to a control classifier instead of applying pairwise comparisons; several alternative procedures have ben proposed to conduct this kind post-hoc test, such as the Bonferroni test [32]. However, in our case a comparison is required to be conducted between several proposed techniques, strategies, and mechanisms. There are also several post-hoc tests that can be adopted for this purpose, however the Nemenyi test was adopted [75]. According to the Nemenyi post-hoc test, two classifiers are significantly different if the difference of their average

[8]Thus allowing us to reject the null hypothesis that assumes that the performance of all the considered classifiers is the same and any differences in their performance is random.

rank is greater than or equal to some critical difference calculated using Equation 2.2 [25]:

$$CD = q_\alpha \sqrt{\frac{k\,(k+1)}{6N}} \tag{2.2}$$

where q_α is the critical value (which can be obtained from any text book on statistics), k is the number of the classifiers, and N is the number of samples (datasets) considered in the comparison. It is interesting to note that the Friedman test might predict a significant difference between several classifiers, however the post-hoc test might fail to identify it. According to Demsar, this is because of the "recognised weaknesses" of the post-hoc tests[9].

The results reported on later in this thesis are compared using the Wilcoxon signed rank test in cases where two classification models are to be compared, and the Friedman test with the Nemenyi post-hoc test in the case of multiple classifier comparison. Both the Wilcoxon and Friedman tests are available in $SPSS$, however no corresponding post-hoc test for the Friedman test is available in $SPSS$. Consequently this was calculated manually using Equation 2.2.

2.8 Terminology

In this section an explanation of the most significant terminology used later in this thesis is provided. Commencing with terminology used with respect to the proposed data distribution techniques used to distribute class labels between nodes within the binary tree hierarchical classification model, and following on with more general terminology. Three different techniques were considered:

1. K-means clustering. This technique uses the well known k-means clustering algorithm to cluster the data into two clusters ($k = 2$) without eliminating any resulting overlap between the classes such as in the case of the mechanism adopted in BCT-OB discussed earlier in section 2.4 where a cleaning mechanism was adopted to reduce the number of overlapping classes between the two resulting clusters. The conjecture advantage is that the presence of overlapping could mitigate against the early mis-classification issue associated with the hierarchical structure.

2. *Divisive* hierarchical clustering. This technique adopts a top down "divisive" hierarchical clustering algorithm to cluster the data whereby clusters are repeatedly divided into sub clusters until some stopping criteria is reached. Again, as in the case of the k-means clustering, overlapping is maintained between classes and no cleaning mechanism was adopted to eliminate it.

[9]Demsar describes this as follows: "Sometimes the Friedman test reports a significant difference but the post-hoc test fails to detect it. This is due to the lower power of the latter. No other conclusion than that some algorithms do differ can be drawn in this case" [25].

3. Data splitting. This technique comprises a simple "cut" of the data into two groups so that each contains a disjoint subset of the entire set of class labels. More specifically, the data splitting technique splits the data into two disjoint groups without overlap between class labels and without taking into account any similarity between them. For example given a data set with six class labels $\{a, b, c, d, e, f\}$, the data splitting might split these into $\{a, b, c\}$ and $\{d, e, f\}$ or any other possible balanced split.

In addition, the following terminology used throughout the thesis:

1. Bagging. This term refers to the process of dividing a given dataset into disjoint partitions and learning a classifier for each partition. In other words this refers to one of the "Bagging-Like-Strategies" explained earlier in section 2.4. Efficiency is the main reason behind the selection of this type of sampling. With respect to the work presented in this thesis tree partitions were used. The reason behind adopting three partitions, was so as to avoid the insufficient information issue associated with sampling with partitioning that increases as the number of partitions increases.

2. Generation time. This term refers to the time required to train a classifier using a given training set comprised of a set of examples.

3. Classification time. This term refers to the time required to classify the examples in a given test set.

4. Directed Acyclic Graph (DAG). This term refers to a graph that comprises a set of nodes (vertices) and directed edges (arcs), where every node has at least one inward or outward edge connecting it to another node in such a way that there are no cycles, in other words there is no sequence of edges starting from a node N that eventually loops back to N [21, 95].

5. *Rooted* DAG. This term refers to a DAG that has exactly one node designated as a root node, a node that has no edges pointing in to it; in other words there is only one node that has no predecessor nodes [81].

2.9 Summary

In summary, this chapter has presented a literature review of the most widely used approaches to solve the multi-class classification problem, namely:

1. Stand-alone classification.

2. Collections of binary classifiers. From the literature, the most significant work is OVO.

3. Ensemble classifiers arranged in: (i) concurrent, (ii) sequential, (iii) binary tree hierarchical form and (iv) DAG hierarchical form.

In the context of binary hierarchical ensemble classification, the previous work can be categorised into two main categories with respect to the adopted technique to distribute class labels between nodes within the binary tree: (i) overlapping between classes at nodes and (ii) no−overlapping between classes at nodes. The work presented in this thesis considered both categories. The previous work on DAG hierarchical ensemble classification has focused on using binary classifiers at nodes rather than groups of classes as proposed in this thesis. As noted earlier in this chapter this can be considered to be a special case of using a set of binary classifiers to solve the multi-class classification problem. Also it can be considered as a way to combine the results from OVO decomposition.

In addition an overview of clustering algorithms has been presented. The reason for this is that clustering algorithms were utilised to distribute classes between nodes within the hierarchy with respect to the work presented later in this thesis. Also a general overview of the statistical tests used latter in this thesis to compare the different classification models was provided. In the following chapter (Chapter 3) the hierarchical ensemble classification model for multi-class classification based on the usage of a Binary Tree (BT) structure will be presented.

Chapter 3

The Binary Tree Hierarchical Classification Model

3.1 Introduction

The nature of the proposed Binary Tree hierarchical classification approach is presented in this chapter. As noted earlier in the introduction to this thesis, the binary tree hierarchical classifier is a form of ensemble classifier. Each node in the hierarchy holds a classifier. Classifiers at the leaves conduct fine-grained (binary) classifications while the classifiers at non-leaf nodes further up the hierarchy conduct coarse-grained classification directed at categorising examples using groups of labels. To remind the reader of the the Binary Tree hierarchy Figure 2.1 from Chapter 2 is given again in Figure 3.1. At the root we classify into two groups of class labels $\{a, b, c, d\}$ and $\{e, f, g\}$. At the next level we split into smaller groups, and so on till we reach classifiers that can associate single class labels with examples. Note that Figure 2.1 is just an example of the proposed hierarchical model; non-leaf child nodes may end up with overlapping classifications because the adopted clustering algorithms may assign examples belonging to the same class to different clusters. Recall from Chapter 2 that there has been some previous work on binary tree based hierarchical ensemble classification (Section 2.4).

The challenges of hierarchical single-label classification, as conceived in this thesis are: (i) how best to distribute (organise) the class labels between nodes so as to produce a Binary Tree classifier that generates the most effective classifications and (ii) how to address the successive mis-classification issue imposed by the hierarchical structure. To address the first issue this chapter reports on several techniques considered to organise (group) the class labels so as to produce a hierarchy that generates an effective classification. These were founded on ideas concerned with the use of clustering and splitting techniques to distribute the class labels as noted in Section 2.4. With respect to the second issue a Multiple Path strategy was proposed (facilitated by the probability or confidence values generated by Naive Bayes and CARM classifiers respectively hosted at the Binary Tree nodes). The first issue is discussed further in Section 3.2 where the

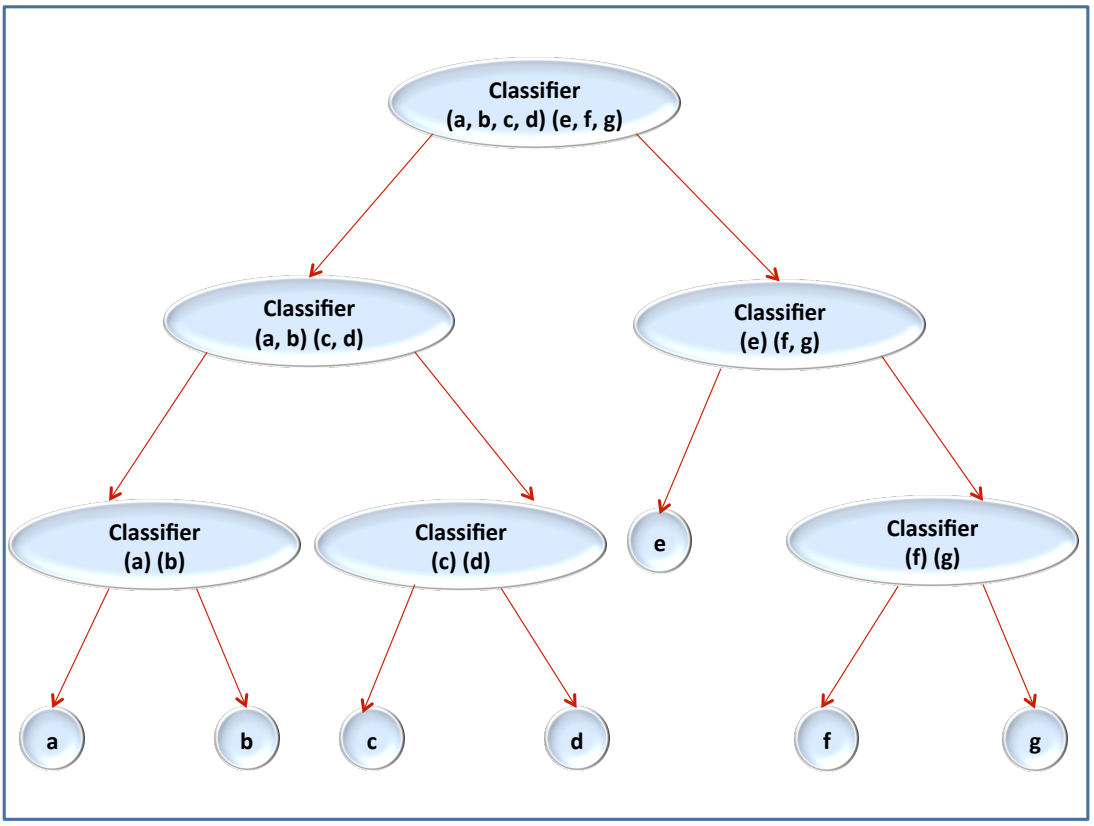


FIGURE 3.1: Binary Tree Hierarchy example.

generation of the DAG ensemble approach is presented in detail. While the second issue is addressed in Section 3.3 where the operation of the proposed approach is presented. Section 3.4 presents an overview of the conducted experiments and the obtained results. Finally, a summary of this chapter is presented in Section 3.5.

3.2 Binary Tree Hierarchical Model Generation

In this section the generation of the proposed Binary Tree hierarchical classification model is explained in detail. Recall that in the proposed model classifiers nearer the root of the hierarchy conduct coarse-grain classification with respect to subsets of the available set of classes. Classifiers at the leaves of the hierarchy conduct fine-grain (binary) classification. To create the hierarchy a classifier needs to be generated for each node in the hierarchy using an appropriately configured training set.

Two classification styles were considered with respect to the nodes in the proposed binary tree ensemble hierarchy: (i) straight forward single "stand-alone" classifiers (Figure 3.2(a)) and (ii) Bagging ensembles (Figure 3.2(b)). With respect to the first style, a simple classifier (of any form Decision Tree, Naive Bayes, or CARM) was generated for each node in the hierarchy. With respect to Bagging, the data set D associated with each node was randomly divided into N disjoint partitions and a classifier generated for each (in the evaluation reported in Section 3.4, $N = 3$ was used).

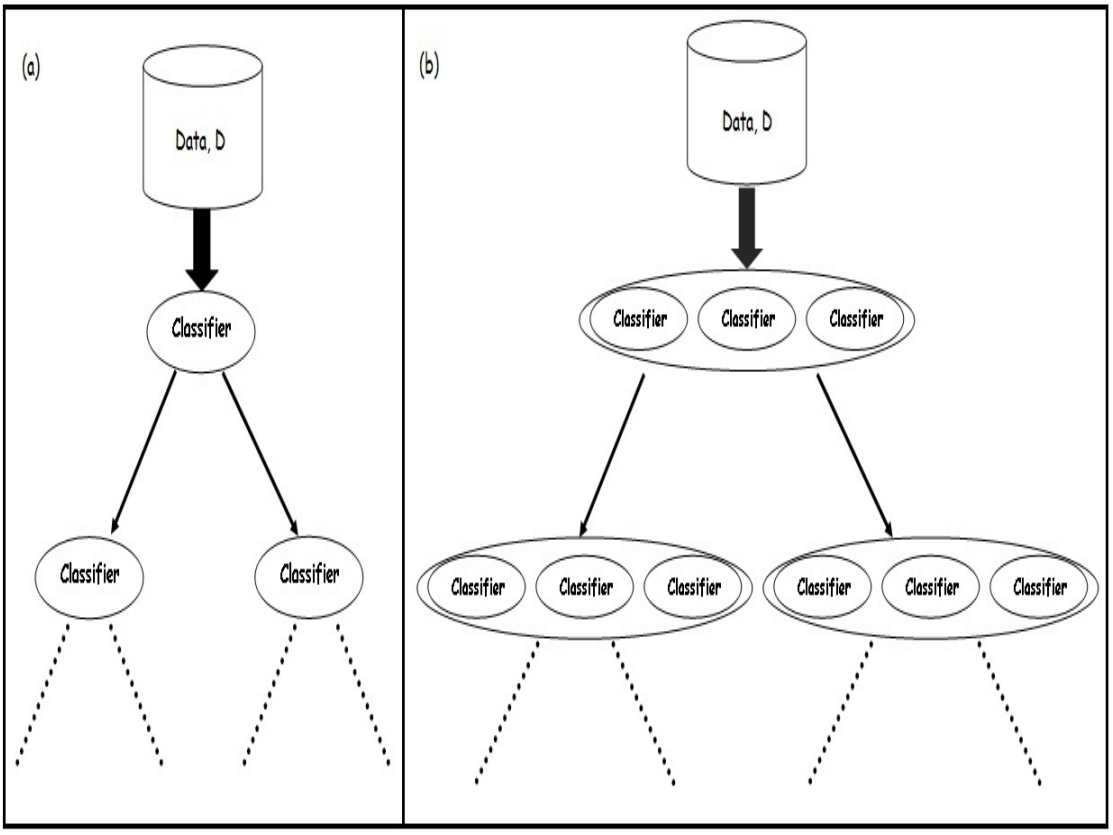


FIGURE 3.2: Binary Tree hierarchical classification model, **(a)** using a single classifier at each node and **(b)** using a Bagging ensemble at each node.

In order to group (divide) the input data D during the hierarchy generation process, three different distribution techniques were considered: (i) k-means clustering, (ii) divisive hierarchical clustering and (iii) data splitting. K-means and *divisive* hierarchical clustering were both described in Section 2.6 in Chapter 2. Among these k-means is the most commonly used partitioning method where examples are divided into k partitions (in our model $k = 2$ was used because of the binary nature of our hierarchies). Hierarchical clustering creates a hierarchical decomposition of the given data. In the context of the work described in this thesis a "divisive" hierarchical clustering (top-down) was used because this fits well with respect to the vision of hierarchical ensemble classification presented in this thesis. Recall from Chapter 2 that the process commences with all examples in one cluster, on each successive iteration, a cluster is split into smaller clusters until a "best" cluster configuration is arrived at (measured using cluster cohesion and separation measures). The idea behind the use of clustering algorithms is, at each level and branch of the hierarchy, to group the available class labels into two disjoint groups (clusters) so that the classes within each group share some similar characteristics. Data splitting comprises a simple "cut" of the data into two groups so that each contains a disjoint subset of the entire set of class labels. More specifically, the data splitting technique split data into two disjoint groups without overlapping between class labels and without taking into account any similarity between them. For example given a data

set with six class labels $\{a, b, c, d, e, f\}$, the data splitting might split these into $\{a, b, c\}$ and $\{d, e, f\}$ or any other possible balanced split.

The proposed Binary Tree hierarchy generation algorithm is presented in Algorithm 1. The algorithm assumes a data structure, called *hierarchy*, comprised of the following fields:

1. Classifier: A classifier at each tree node.
2. Left: Reference to left branch of the hierarchy (root and body nodes only, set to null at leaves).
3. Right: Reference to right branch of the hierarchy (root and body nodes only, set to null at leaves).

Considering the algorithm presented in Algorithm 1 in further detail. The *Generate_Hierarchy* procedure is recursive. On each recursion the input to the *Generate_Hierarchy* procedure is the data set D (initially this is the entire training set). If the number of classes featured in D is two, a binary classifier is constructed (to distinguish between the two classes) (lines 15-17). The most sophisticated part of the *Generate_Hierarchy* procedure is where the number of classes featured in D is more than two. In this case the examples in D are divided into two groups $D1$ and $D2$ each with a meta-class label, $K1$ and $K2$, associated with it (line 20). A Classifier is then constructed to discriminate between $K1$ and $K2$ (line 21). The *Generate_Hierarchy* procedure is then called again, once with $D1$ (representing the right branch of the hierarchy) if the number of classes featured in $D1$ is more than one class, and once with $D2$ (representing the left branch of the hierarchy) if the number of classes featured in $D2$ is more than one class (lines 26 and 32).

It is interesting to note that if the clustering algorithms, k-mean or *divisive* hierarchical clustering, are used to divide class labels between nodes during the generation process; the number of classifiers that will be generated cannot be calculated in advance. While if a data splitting technique is used during the hierarchy generation process, the number of classifiers needed to be trained is $N-1$, where N is the number of class labels in a given dataset.

3.3 Binary Tree Hierarchical Model Operation

Section 3.2 explained the generation of the proposed Binary Tree hierarchical classification model. After the model has been generated the intention is to use it to classify new unseen data examples. In this section the process whereby this is achieved is explained. Two strategies were considered for classifying individual examples: (i) Single Path and (ii) Multiple Path. The Single Path strategy is the most straightforward with which to classify a new example whereby the aim is to identify a single path leading through the hierarchy, as dictated by the node classifiers, until a leaf node associated with a

Algorithm 1 Binary Tree Hierarchy Generation

INPUT
D, the input training dataset
Classification method, procedure for building the classifiers at the hierarchy nodes
Clustering (or splitting) technique, procedure for splitting the classes between nodes
OUTPUT
The generated Binary Tree Hierarchy

Start
$root$ = the root node for the Binary Tree
$root = Generate\_Hierarchy(D)$
End

function $Generate\_Hierarchy(D)$
    create a hierarchy node N;
    if number of classes featured in $D == 2$ **then**
        create a hierarchy $classifier$ (using classification method), to distinguish between the two (real) classes (binary classification)
        assign null value to hierarchy $left$, and $right$
    else
        cluster D into two clusters $K1$ and $K2$ (using clustering or splitting)
        recast labels in D so that they correspond to $K1$ and $K2$;
        create a hierarchy $classifier$ (using classification method), to distinguish between $K1$ and $K2$;
        $D1$ = examples in D containing class labels in $K1$;
        if number of classes featured in $D1 == 1$ **then**
            assign null value to hierarchy $right$
        else
            hierarchy $right$ = Generate_ Hierarchy ($D1$);
        end if
        $D2$ = examples in D containing class labels in $K2$;
        if number of classes featured in $D2 == 1$ **then**
            assign null value to hierarchy $left$
        else
            hierarchy $left$ = Generate_ Hierarchy ($D2$);
        end if
    end if
    return N;
end function

single class label is arrived at. However, this strategy does not address the issue that if a example is mis-classified early on in the process it will continue to be mis-classified later on in the process. The second strategy attempts to address this issue by allowing more than one path to be followed. The Multiple Path strategy was realised by utilising Naive Bayes classifiers and Classification Association Rule Miners (CARM), which feature respectively probability and confidence values that can be used to determine where single or multiple paths should be followed. More specifically, more than one path was followed within the hierarchy according to a predefined threshold σ, in the case of Naive Bayes classifiers $0 \leq \sigma < 1$, while in the case of Classification Association Rule Miners (CARM) $0 \leq \sigma \leq 100$.

A further issue that results when more than one path is followed through the hierarchy is that more than one final "candidate class" label may be arrived at, the question then is which class label to select? Three different mechanisms were suggested to determine the final resulting class label: (i) simply selecting the candidate class associated with the highest "individual" probability (or confidence) value, (ii) generating an *accumulated weight* for each candidate class and selecting the class associated with the highest accumulated weight or (iii) applying some Voting scheme and selecting the candidate class associated with the highest vote. Thus we have three variations of the Multiple Path strategy: (i) Multiple Path with Best Individual Probability/Confidence (BIP/BIC) class label selection, (ii) Multiple Path with Normalised Accumulated Probability/Confidence (NAP/NAC) class label selection and (iii) Multiple Path with Voting class label selection

The rest of this section is organised as follows: Sub-section 3.3.1 explains the Single Path strategy, while Sub-section 3.3.2 considers the Multiple Path strategy.

3.3.1 Single Path Strategy

In the Single Path strategy only one path will be followed according to the classification at each hierarchy node. Recall from the above that during the generation process sets of class labels are grouped. For simplicity, and in acknowledgement of the binary nature of our example hierarchies, we refer to these groups as the $left$ and $right$ groups. The procedure for using the hierarchy to classify an example, e, is summarised in Algorithm 2. The $BinaryTreeSinglePathClassify$ procedure is recursive. On each recursion the algorithm is called with two parameters: (i) e, the example to be classified and (ii) a pointer to the current node location in the hierarchy (at start this will be the root node). How the process proceeds then depends on the nature of the class label returned by the classifier at the current node in the hierarchy. If the returned class belongs to one of either the $right$ or $left$ groups the $BinaryTreeSinglePathClassify$ procedure will be called again with the parameters: (i) either the left or right child node as appropriate, and of course (ii) the example e (lines 17 and 19). If the returned class label is a specific class label (as opposed to some grouping of labels) this class label will be returned as the label to be associated with the given example and the algorithm terminated (lines 13 and 14).

In the same way that it is not possible to calculate how many nodes there will be in the binary tree prior to generation of the tree when using clustering (either $k-means$ or *divisive* hierarchical clustering) to distribute class labels between nodes; it was not possible to calculate in advance of generating the tree the number of classifiers that will need to be evaluated in order to classify a new example. While when a data splitting technique is used during the hierarchy generation process, the number of classifiers that will need to be evaluated when only a single path is followed within the hierarchy will be, in the worst case, $\log_2 N$ (the depth of the tree); where N is the number of class labels in a given dataset.

Algorithm 2 Binary Tree Hierarchy Single Path Classification

```
1: INPUT
2: e = A new unseen example
3: N a pointer to the current node in the hierarchy (root node at start)
4: OUTPUT
5: The predicted class label c for the input example e
6:
7: Start
8: c = BinaryTreeSinglePathClassify(e, N)
9: End
10:
11: function BinaryTreeSinglePathClassify(e, N)
12:     C = Classification result for e using classifier at node N
13:     if |C| == 1 then
14:         return C
15:     else
16:         if C ∈ N.rightClassGroup then
17:             return (BinaryTreeSinglePathClassify(e, N.rightBranch))
18:         else
19:             return (BinaryTreeSinglePathClassify(e, N.leftBranch))
20:         end if
21:     end if
22: end function
```

3.3.2 Multiple Path Strategy

As already noted, a possible issue with the single path strategy is that if a mis-classification occurs early on in the process there is no opportunity for rectifying this situation later on in the process. To address this problem a Multiple Path strategy was proposed. As mentioned earlier, Naive Bayes classifiers and Classification Association Rule Miners (CARM) were used, so that the Bayesian probability p (or confidence value) associated with the individual class groups, at each tree node, could be used to dictate whether one or two branches will be followed according to a predefined threshold σ.

In order to decide the final class label from the collection of "candidate classes" resulting from following multiple paths, three different mechanisms were suggested to determining the final resulting class label: (i) Best Individual Probability/Confidence (BIP/BIC), (ii) Normalised Accumulated Probability/Confidence (NAP/NAC) and (iii) Voting. Using the BIP or BIC mechanisms the "individual" probability (confidence) values associated with the identified candidate classes at the leaf nodes were used to select a final class label. Using the NAP or NAC mechanism all probability (confidence) values in a followed path are taken into consideration to produce an accumulated value. Using the Voting mechanism, the number of votes for each candidate class is calculated and the candidate class associated with the highest vote will be assigned as the class label for the given example.

The remainder of this sub-section is organised as follows: Sub-section 3.3.2 considers the multiple path strategy when using CARM classifiers at the nodes in the binary hierarchy, while Sub-section 3.3.2 considers the multiple path strategy when using Naive Bayes classifiers at the nodes in the binary hierarchy.

The Multiple Path Strategy Using CARM Classifiers at Nodes

Using the confidence values generated by CARM classifiers to follow multiple paths within the hierarchy, the confidence value $Conf$ associated with a class group ($Conf_{N.leftClassGroup}$ or $Conf_{N.rightClassGroup}$) is used to indicate whether one or two branches (due to the binary structure of our hierarchy) will be followed using the proposed σ threshold. If the $Conf$ value of the branch associated with the highest confidence is less than σ both branches emanating from the node will be explored further, otherwise the branch with the highest associated $Conf$ value will be selected.

With respect to the above it should be recalled that a classifier generated using a CARM algorithm comprises a set of CARs whereby the CARs are typically ordered according to confidence value. CARs with the highest confidence are listed first. If two CARs have the same confidence usually the more general rule (that with the smallest antecedent) will appear first, with more specific rules appearing later[1]. Typically the classifier will also include a default rule to be fired when no other rule fits the given example, which will return the most frequently occurring class label in the original training set. Given a new example to be classified, the first rule whose antecedent matches the example (or the default rule) is used to classify the example. However, our Multiple Path strategy requires that we have confidence values for both branches emanating from a node. Thus the CARM classifiers used were modified so that, where possible, the confidence values associated with both branches were returned by finding the first rule in the rule base with respect to both classes (where such rules existed).

When using the Multiple Path strategy coupled with the (BIC) mechanism the suggested procedure is presented in Algorithm 3. The algorithm is similar to the proposed Single Path strategy algorithm (Algorithm 2) except with respect to the use of the σ threshold to decide whether to follow a single branch or both branches emanating from a node. At the end of the algorithm a list L, which holds all the identified potential class labels with their associated confidence values for the given case, is processed to select the class label c with the highest confidence value. The procedure $MultiPathBestConf(e, N)$ is called recursively as the process progresses. On each recursion the CARM classifier held at the current node is used to produce a confidence value (line 8), with respect to e for the $leftClassGroup$ and the $rightClassGroup$. We then follow one or two branches according to the relative nature of the confidence values returned using the CARM classifier at the current node and the σ threshold. Whenever the size of a class group considered at a node is equal to one, indicating that the group comprises a single class label, the class label and associated confidence value are added to L. At the end of the process (line 36) L is processed to identify the class label with the highest associated confidence value.

Algorithm 4 presents the Multiple Path strategy coupled with the Normalised Accumulated Confidence (NAC) mechanism. The main difference between the BIC and NAC mechanisms is that for the latter all confidence values are stored with respect to each

[1]Some authors argue that the more specific rule should be listed first, this remains an open question.

Algorithm 3 Multiple Path Classification Coupled with BIC

INPUT
e a new unseen example
N a pointer to the current node in the hierarchy (root node at start)
OUTPUT
c the predicted class label of the input example e

L the set of class labels, together with their associated confidence values, maintained as the procedure progresses, set to {} at start

START PROCEDURE $MultiPathBestConf(e, N)$
C = Class label set for e with the associated confidence values ($Conf$) generated using classifier held at node N ($C = \{N.leftClassGroup, N.rightClassGroup\}$)
if (($Conf_{(N.leftClassGroup)} > Conf_{(N.rightClassGroup)}$ and ($Conf_{(N.leftClassGroup)} > \sigma$)) **then**
    if ($|N.leftClassGroup| == 1$) **then**
        Add class label c_i ($c \in N.leftClassGroup$) to class list L with $Conf_{c_i}$
    else
        $MultiPathBestConf(e, N.leftBranch)$
    end if
else
    if (($Conf_{(N.rightClassGroup)} > Conf_{(N.leftClassGroup)}$ and ($Conf_{(N.rightClassGroup)} > \sigma$)) **then**
        if ($|N.rightClassGroup| == 1$) **then**
            Add class label c_i ($c \in N.rightClassGroup$) to class list L with $Conf_{c_i}$
        else
            $MultiPathBestConf(e, N.rightBranch)$
        end if
    else
        if ($|N.leftClassGroup| == 1$) **then**
            Add class label c_i ($c \in N.leftClassGroup$) to class list L with $Conf_{c_i}$
        else
            $MultiPathBestConf(e, N.leftBranch)$
        end if
        if ($|N.rightClassGroup| == 1$) **then**
            Add class label c_i ($c \in N.rightClassGroup$) to class list L with $Conf_{c_i}$
        else
            $MultiPathBestConf(e, N.rightBranch)$
        end if
    end if
end if
END PROCEDURE $MultiPathBestConf(e, N)$

Process L and select class label c with highest confidence

path followed (not just the confidence values at the leaf nodes). Consequently a weighting may be derived for each candidate class. We refer to this weighting as the Normalised Accumulated Confidence ($NormalisedAccumulatedConf$) value. In order to determine the Normalised Accumulated Confidence value associated with a path two additional parameters (in addition to the parameters in Algorithm 3) are used: $AccumConf$ and $ConfCount$, where $AccumConf$ is used to store the summation of the confidence values for the path followed, while $ConfCount$ is used to store the number of confidence

values for the path followed (so that the final accumulated confidence can be normalised).

More specifically, the Normalised Accumulated Confidence value, which will be associated with each candidate class label, is calculated as follows:

$$NormalisedAccumConf = AccumConf \div Confcount \tag{3.1}$$

where $0 < NormalisedConf \leq 100$.

As noted above the generated CARM classifiers held at nodes were modified to return the confidence value associated with both class labels represented by each node. However, in some cases it was not possible to identify both confidence values (no rule in the rule base matched the second class with respect to the given example). In this case only the single identified confidence value was used when calculating the $NormalisedAccumulatedConfidence$ value for a specific path (See Algorithm 4, lines 31-35 and 37-41). If the path that did not feature a confidence value was followed, no confidence value was assumed. Note here that the Null in Algorithm 4 refers to unknown confidence value. It is worth noting here that the reason for setting the first condition ($Conf(_{N.leftClassGroup}) > Conf(_{N.rightClassGroup})$) in Algorithms 3 and 4 was again that the confidence values cannot always be determined for one of the branches. More specifically, if the $Conf$ value of the branch associated with the highest confidence (or the known confidence) is less than σ both branches emanating from the node will be explored further, otherwise the branch with the highest associated $Conf$ value will be selected.

With respect to the Multiple Path strategy coupled with the Voting class label selection mechanism, the algorithm is very similar to Algorithms 3 and 4, but much simpler as there is no need to store confidence values during the classification process. Only the individual class labels obtained during traversal of the BT will be added to L. On completion L will contain a set of candidate class labels, the class label with highest occurrences count in L will be assigned to the example under consideration.

The Multiple Path Strategy Using Naive Bayes Classifiers at Nodes

With respect to utilising Naive Bayes probability values to follow multiple paths within the binary tree hierarchy, in a similar manner to that when using confidence values generated by CARM classifiers to follow multiple paths within the hierarchy, the Bayesian probability P associated with individual class groups ($P_{N.leftClassGroup}$ and $P_{N.rightClassGroup}$) was used to dictate whether one or two branches (because of the binary nature of our hierarchy) will be followed according to the predefined threshold sigma (σ). If $P_{N.leftClassGroup} > \sigma$ and $P_{N.rightClassGroup} > \sigma$ then both branches will be explored, otherwise the branch with the highest associated P value will be selected.

The Multiple Path BIP algorithm is presented in Algorithm 5. While Algorithm 6 presents the multiple path strategy coupled with normalised accumulated probability.

Algorithm 4 Multiple Path Coupled with NAC

INPUT
e a new unseen example
N a pointer to the current node in the hierarchy (root node at start)
$AccumConf$ the Accumulated summation of the confidence values in the followed path (initially 0.0)
$ConfCount$ Confidence counter keeping the number of confidence values in the followed path
OUTPUT
c the predicted class label of the input example e
L the set of class labels, together with their associated normalised accumulated confidence values, maintained as the procedure progresses, set to {} at start
START PROCEDURE $MultiPathNormalisedConf(e, N, AccumConf, ConfCount)$
C = Class label set for e with the associated confidence values ($Conf$) generated using classifier held at node N ($C = \{N.leftClassGroup, N.rightClassGroup\}$)
if ($Conf_{(N.leftClassGroup)} > Conf_{(N.rightClassGroup)}$ and ($Conf_{(N.leftClassGroup)} > \sigma$)) **then**
    $leftAccumConf = Conf_{(N.leftClassGroup)} + AccumConf$
    $leftConfCount = ConfCount + 1$
    if ($|N.leftClassGroup| == 1$) **then**
        $leftNormalisedAccumConf = leftAccumConf/leftConfCount$
        Add class label c_i to class list L with the $leftNormalisedAccumConf$.
    else
        $MultiPathNormalisedConf(e, N.leftBranch, leftAccumConf, leftConfCount)$
    end if
else
    if ($Conf_{(N.rightClassGroup)} > Conf_{(N.leftClassGroup)}$ and ($Conf_{(N.rightClassGroup)} > \sigma$)) **then**
    $rightAccumConf = Conf_{(N.rightClassGroup)} + AccumConf$
    $rightConfCount = ConfCount + 1$
        if ($|N.rightClassGroup| == 1$) **then**
        $rightNormalisedAccumConf = rightAccumConf/rightConfCount$
        Add class label c_i to class list L with the $rightNormalisedAccumConf$.
        else
        $MultiPathNormalisedConf(e, N.rightBranch, rightAccumConf, rightConfCount)$
        end if
    else
        if ($Conf_{(N.leftClassGroup)}$!= Null) **then**
        $leftAccumConf = Conf_{(N.leftClassGroup)} + AccumConf$
        $leftConfCount = ConfCount + 1$
        else
        $leftAccumConf = AccumConf$, $leftConfCount = ConfCount$
        end if
        if ($Conf_{(N.rightClassGroup)}$!= Null) **then**
        $rightAccumConf = Conf_{(N.rightClassGroup)} + AccumConf$
        $rightConfCount = ConfCount + 1$
        else
        $rightAccumConf = AccumConf$, $rightConfCount = ConfCount$
        end if
        if ($|N.leftClassGroup| == 1$) **then**
        $leftNormalisedAccumConf = leftAccumConf/leftConfCount$
        Add class label c_i to class list L with the $leftNormalisedAccumConf$.
        else
        $MultiPathNormalisedConf(e, N.leftBranch, leftAccumConf, leftConfCount)$
        end if

Algorithm 4 Multiple Path Coupled with NAC (Continued)

```
49:         if (|N.rightClassGroup| == 1) then
50:             rightNormalisedAccumConf = rightAccumConf/rightConfCount
51:             Add class label c_i to class list L with the rightNormalisedAccumConf.
52:         else
53:         MultiPathNormalisedConf(e, N.rightBranch, rightAccumConf, rightConfCount)
54:         end if
55:     end if
56: end if
57: END PROCEDURE MultiPathNormalisedConf(e, N, AccumConf, ConfCount)

58: Process L and select class label c with highest normalised accumulated confidence value
```

Algorithm 5 is similar to Algorithm 3, while Algorithm 6 is similar to Algorithm 4 as explained earlier. However, the probability value for both class groups at any node can be identified, unlike in the case when using CARM confidence values where in some cases it was not possible to identify both confidence values.

Again, if the clustering algorithms, $k-$mean or *divisive* hierarchical clustering, are used to divide class labels between nodes during the generation process; the number of classifiers that need to be evaluated in order to classify a new example can not be calculated in advance. While if a data splitting technique is used during the hierarchy generation process, the number of classifiers that need to be evaluated when multiple paths are followed within the hierarchy is $N - 1$ in the worst case (visit all nodes in the tree), where N is the number of class labels in a given dataset.

3.4 Experiments and Results

In this section we present an overview of the adopted experimental set up and the evaluation results obtained. The effectiveness of the suggested Binary Tree hierarchical classification model was evaluated using the fourteen different data sets identified in Chapter 1, and pre-processed using the LUCS-KDD-DN software [23]. Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were average accuracy and average AUC (Area Under the receiver operating Curve). Although the results in terms of average accuracy and average AUC are both included in this section, we will discuss the results in terms of average AUC only because of: (i) the theoretical and empirical evidences that AUC is a better measure than accuracy for evaluating learning algorithms [52] and (ii) the inclusion of unbalanced datasets within the considered evaluation datasets (accuracy does not take class priors into consideration). Note here that the low AUC values associated with the Nursery, Glass, Zoo, and Ecoli data sets are caused, at least partially, by the issue reported earlier in Chapter 1 that when using TCV with respect to data sets that feature a small number of examples for some classes some folds may not include any examples for a particular class. Note that with respect to the discussion of the statistical significance of the results presented in this section details of the calculation of significance is provided in Appendix C.

Algorithm 5 Multiple Path Classification Coupled with BIP

INPUT
e a new unseen example
N a pointer to the current node in the hierarchy (root node at start)
OUTPUT
c the predicted class label of the input example e

START PROCEDURE $MultiPathBestProb(e, N)$
C = Class label set for e with the associated probabilities generated using classifier held at node N ($C = \{N.leftClassGroup, N.rightClassGroup\}$)
if ($P_{N.leftClassGroup} > \sigma$ and $P_{N.righttClassGroup} > \sigma$) **then**
    if ($|N.leftClassGroup| = 1$) **then**
        Add class label c_i ($c \in N.leftClassGroup$) to class list L with prob. P_{c_i}
    else
        $MultiPathBestPro(e, N.leftBranch)$
    end if
    if ($|N.rightClassGroup| = 1$) **then**
        Add class label c_i ($c \in N.rightClassGroup$) to class list L with prob. P_{c_i}
    else
        $MultiPathBestPro(e, N.rightBranch)$
    end if
else
    if ($P_{N.leftClassGroup} > P_{N.rightClassGroup}$) **then**
        if ($|N.leftClassGroup| = 1$) **then**
            Add class label c_i ($c \in N.leftClassGroup$) to class list L with prob. P_{c_i}
        else
            $MultiPathBestProb(e, N.leftBranch)$
        end if
    else
        if ($|N.rightClassGroup| = 1$) **then**
            Add class label c_i ($c \in N.rightClassGroup$) to class list L with prob. P_{c_i}
        else
            $MultiPathBestProb(e, N.rightBranch)$
        end if
    end if
end if
Process L and select class label c with highest probability

For the evaluation three classification algorithms (Decision Tree, Naive Bayes, CARM) and two classification styles (Stand-alone, and Bagging) were used coupled with the three proposed partitioning techniques (k-means, *divisive* hierarchical clustering and splitting technique) and the two proposed classification strategies (Single and Multiple Path). In addition, with respect to the Multiple Path strategy, the three proposed mechanisms for arriving at a final classification decision, BIP (BIC), Voting and NAP (NAC), were considered. Thus, in total, eighteen different Binary Tree hierarchical classification model variations were considered, these are summarised in Table 3.1:
With respect to the Bagging methods three classifiers were generated with respect to each node. Both the Single and Multiple Path classification strategies were considered for each of the above variations.

For comparison purposes alternative forms of classification were also applied to the data sets as follows:

Algorithm 6 Multiple Path Coupled with NAP

INPUT
e a new unseen example
N a pointer to the current node in the hierarchy (root node at start)
$AccumWieght$ the accumulated Weight for the followed path (initially 0.0)
OUTPUT
c the predicted class label of the input example e

START PROCEDURE $MultiPathNormalisedProb(e, N, AccumWieght)$
C = Class label set for e with the associated probabilities generated using classifier held at node N ($C = \{N.leftClassGroup, N.rightClassGroup\}$)
if ($P_{N.leftClassGroup} > \sigma$ and $P_{N.rightClassGroup} > \sigma$) **then**
    $leftAccumWeight = AccumWeight + P_{N.leftClassGroup}$
    $rightAccumWeight = AccumWeight + P_{N.rightClassGroup}$
    if ($|N.leftClassGroup| = 1$) **then**
        Add class label c_i ($c \in N.leftClassGroup$) to class list L with weight.
        $leftAccumWeight$ (after weight normalisation)
    else
        $MultiPathNormalisedProb(e, N.leftBranch, leftAccumWeight)$
    end if
    if ($|N.rightClassGroup| = 1$) **then**
        Add class label c_i ($c \in N.rightClassGroup$) to class list L with weight.
        $rightAccumWeight$ (after weight normalisation)
    else
        $MultiPathNormalisedProb(e, N.rightBranch, rightAccumWeight)$
    end if
else
    if ($P_{N.leftClassGroup} > P_{N.rightClassGroup}$) **then**
        $leftAccumWeight = AccumWeight + P_{N.leftClassGroup}$
        if ($|N.leftClassGroup| = 1$) **then**
            Add class label c_i ($c \in N.leftClassGroup$) to class list L with weight.
            $leftAccumWeight$ (after weight normalisation)
        else
            $MultiPathNormalisedProb(e, N.leftBranch, leftAccumWeight)$
        end if
    else
        $rightAccumWeight = AccumWeight + P_{N.rightClassGroup}$
        if ($|N.rightClassGroup| = 1$) **then**
            Add class label c_i ($c \in N.rightClassGroup$) to class list L with weight.
            $rightAccumWeight$ (after weight normalisation)
        else
            $MultiPathNormalisedProb(e, N.rightBranch, rightAccumWeight)$
        end if
    end if
end if
Process L and select class label c with highest normalised weight

1. A number of "stand alone" classifiers, namely: **Naive Bayes**, **Decision tree**, and **CARM**. Other forms of single classification model could have been selected but Naive Bayes, Decision tree, and CARM were chosen because these were also used in the context of the Binary Tree (BT) model.

2. A **Bagging** ensemble using a combination of three classifiers, again Naive Bayes, Decision tree, and CARM were used as the base classifiers.

TABLE 3.1: Binary Tree hierarchical classification model variations

Classifiers at Nodes	Data distribution Technique		
	K-means	Data Splitting	Hierarchical Clustering
Decision tree	K-means&DT	DS&DT	HC&DT
Decision Tree Bagging	K-means&DTB	DS&DTB	HC&DTB
Naive Bayes	K-means&N	DS&N	HC&N
Naive Bagging	K-means&NB	DS&NB	HC&NB
CARM	K-means&CARM	DS&CARM	HC&CARM
CARM Bagging	K-means&CARMB	DS&CARMB	HC&CARMB

The objectives of the evaluation were as follows:

1. To compare the classification effectiveness of the three proposed data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting.
2. To compare the classification effectiveness of the three considered node classifiers: (i) Decision Tree, (ii) Naive Bayes and (iii) CARM.
3. To compare the classification effectiveness of the two considered classification styles: (i) Stand-alone and (ii) Bagging.
4. To compare the three proposed class label selection methods associated with the Multiple Path strategy: (i) BIP (or BIC), (ii) NAP (or NAC) and (iii) Voting, in terms of classification effectiveness.
5. To compare the use of the Single Path strategy against the use of the Multiple Path strategy for hierarchical ensemble classification.
6. To compare the classification effectiveness of the proposed Binary Tree hierarchical ensemble classification model with respect to stand-alone classification versus alternative established ensemble methods (namely Bagging).
7. To compare the efficiency of the proposed binary tree hierarchical ensemble classifiers in terms of generation time and evaluation time.

The results obtained are presented in the following Sub-sections. These are organised as follows. Sub-section 3.4.1 presents the results obtained using the Single Path strategy with respect to: (i) the three alternative classification algorithms, (ii) the two considered classification styles (stand-alone and bagging) and (iii) the three considered data distribution techniques. Sub-section 3.4.2 presents the results obtained using the Multiple Path strategy with respect to: (i) the two classification algorithms considered (Naive Bayes and CARM), (ii) the three considered data distribution techniques and (iii) the two proposed class label selection methods. Sub-section 3.4.3 then provides a comparison between the Single Path and Multiple Path strategies, whilst Sub-section

3.4.4 considers the results obtained when comparing the usage of conventional methods, stand-alone and bagging with the results obtained from the proposed Binary Tree hierarchical classification model.

In the context of the above listed evaluation objectives, the results in the context of the first three of these are discussed in Sub-sections 3.4.1 and 3.4.2. The reasons for this are that: (i) the comparisons of the different considered distribution techniques, the different considered classifier generators, and the different considered classification styles are required to be conducted with respect to both Single and Multiple Path strategies; (ii) to consider the settings associated with the Multiple Path strategy before commencing to compare the different classifiers, distribution techniques or classification styles; (iii) to avoid repetition of the presented results; and (iv) to maintain consistency with the way that evaluation results are presented in later chapters (these are all presented in a similar manner as in this chapter). The results in the context of the fourth, fifth and sixth evaluation objective are discussed in Sub-sections 3.4.2, 3.4.3 and 3.4.4 respectively. With respect to the last evaluation objective, the comparative efficiency of the proposed binary tree hierarchical ensemble classifiers, this is considered throughout the results sub-sections.

3.4.1 Single Path Experiments and Results

This section presents the results obtained using the Single Path strategy for the different Binary Tree hierarchical classification model variations. More specifically, in this chapter a comparison between: (i) the different considered distribution techniques (k-means, *divisive* hierarchical clustering, and data splitting), (ii) the different considered classifier generators (Decision Tree, Naive Bayes and CARM) and (iii) the different considered classification styles (Stand-alone and Bagging), with respect to the Single Path strategy is presented. First the results obtained from each classifier, in stand-alone or Bagging mode, with respect to different data distribution techniques are presented. Consequently, a comparison of the different distribution techniques, and the two classification styles with respect to each of the considered classifiers is presented. Then a comparison between the different considered classifiers is conducted.

Commencing with the results obtained when using Decision Tree classifiers at each tree node, both stand-alone decision tree and bagging of decision trees, coupled with the three alternative data distribution techniques. Table 3.2 presents the obtained results in terms of average accuracy, and average AUC (best results highlighted in bold font). From the table it can be clearly observed that usage of a single (stand-alone) decision tree classifier at each tree node significantly outperforms usage bagging at each node. The reason behind the weakness of bagging is the insufficient information for training the classifiers as we drill down in the tree. Recall that disjoint partitions were used to assign samples for the base classifiers within the Bagging, the reason behind choosing this form of sampling was to limit the complexity of the proposed model. More specifically, using other forms of sampling such as sampling with replacement, will result in a higher

generation and classification times. With respect to comparing the three considered data distribution techniques, it can be noted that the results of k-means and data splitting are very close and outperform hierarchical clustering. The reason behind the weakness of the hierarchical clustering, especially when dealing with large data sets, was that redundant clusters were generated at the first levels in the tree where high overlap of class labels occurred between clusters, as a consequence of which: (i) more levels where generated with respect to the resulting tree, thus the likelihood of mis-classification increased during the classification stage and (ii) it became harder for individual classifier to distinguish between the groups of class labels. The best overall results, in terms of average AUC, were obtained when stand-alone Decision Tree classifiers were generated at each tree node coupled with usage of the data splitting technique for distributing data between nodes within the hierarchy. With respect to the statistical evaluation of the results obtained from the three considered data distribution techniques, Friedman test indicated that there was a statistically significant difference between the three considered techniques, however, Nemenyi test failed to detect any significant differences between them. Further information, with respect to the conducted statistical tests is included in Appendix C.

The results obtained with respect to the run-time experiments for both Decision Tree classifiers and Bagging of decision tree classifiers at nodes coupled with the three considered data distribution techniques, are presented in Table 3.3. The table includes the generation (training) and classification times for each. With respect to the data distribution techniques it can be observed that the lowest generation and classification times resulted when using data splitting, followed by k-means and hierarchical clustering respectively. In addition, it can be observed that the lowest generation and classification times were obtained when using Bagging of decision trees at nodes and data splitting, to generate the binary tree hierarchical classification model. Note here that disjoint partitions were used with respect to the generated samples using for the bagging, however, using another sampling technique would result in higher run times.

With respect to the results obtained when using Naive Bayes classifiers at each tree node, both stand-alone Naive Bayes and Bagging of Naive Bayes, coupled with the three proposed data distribution techniques, Table 3.4 presents the obtained results in terms of average accuracy, and average AUC (best results highlighted in bold font). From the table it can be clearly observed, as in the case of the Decision Tree hierarchies discussed above, that using single (stand-alone) Naive Bayes classifiers at each tree node significantly outperforms using Bagging of Naive Bayes at each node. Comparing the three considered data distribution techniques, it can be noted that the results of k-means and data splitting are similar, both clearly outperformed the usage of hierarchical clustering. According to the statistical evaluation, k-means and data splitting performed statistically better than hierarchical clustering. While the performance of k-means and data splitting was not statistically different.

TABLE 3.2: Average Accuracy and AUC values obtained using Decision Tree classifiers and Bagging of Decision Trees classifiers at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting

Data set	Classes	K-means&DT		DS&DT		HC&DT		K-means&DTB		DS&DTB		HC&DTB	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
WaveForm	3	**59.68**	**0.60**	58.56	0.59	58.98	0.59	58.08	0.58	57.48	0.58	59.44	0.59
Wine	3	68.75	0.67	65.22	0.64	**70.54**	**0.68**	64.05	0.64	62.45	0.60	60.09	0.59
Nursery	5	35.69	0.19	33.51	0.27	59.70	0.32	**86.20**	**0.46**	64.55	0.32	67.32	0.36
Heart	5	47.97	0.31	49.89	**0.32**	47.48	0.26	47.41	0.24	51.42	0.28	**52.86**	0.25
PageBlocks	5	**92.56**	**0.49**	92.40	0.48	74.28	0.36	89.77	0.20	91.56	0.28	72.88	0.20
Dermatology	6	58.90	0.55	**61.37**	**0.60**	60.25	0.54	51.90	0.43	54.54	0.49	47.30	0.42
Glass	7	60.44	**0.39**	**60.91**	0.38	56.95	0.34	52.10	0.28	57.43	0.28	52.90	0.25
Zoo	7	**85.00**	**0.50**	**85.00**	**0.50**	82.09	0.51	76.27	0.44	71.36	0.41	75.45	0.43
Ecoli	8	76.90	0.33	**78.72**	**0.35**	56.29	0.28	63.59	0.23	71.63	0.23	49.44	0.22
Led	10	**75.00**	**0.75**	**75.00**	**0.75**	59.94	0.59	71.75	0.72	73.22	0.73	51.75	0.51
PenDigits	10	**67.30**	**0.67**	66.78	**0.67**	66.30	0.66	67.22	**0.67**	66.74	**0.67**	59.44	0.59
Soybean	15	**64.62**	**0.65**	63.02	0.56	52.67	0.47	48.74	0.38	59.06	0.49	48.01	0.39
ChessKRVK	18	25.97	0.21	**27.41**	**0.25**	24.67	0.18	30.04	0.20	29.98	0.17	25.02	0.15
LetRecog	26	**42.99**	**0.43**	42.76	**0.43**	38.59	0.39	31.12	0.31	36.92	0.37	24.26	0.24
Mean		**61.56**	0.48	61.47	**0.49**	57.77	0.44	59.87	0.41	60.60	0.42	53.30	0.37

TABLE 3.3: Run time results (in seconds) obtained using Decision Tree classifiers and Bagging of Decision Tree classifiers at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	Generation Time						Classification Time					
	K-means DT	DS& DT	HC& DT	K-means& DTB	DS& DTB	HC& DTB	K-means& DT	DS& DT	HC& DT	K-means& DTB	DS& DTB	HC& DTB
Waveform	2.255	1.213	3.816	1.890	1.080	3.127	0.390	0.388	0.064	0.043	0.436	0.417
Wine	0.181	0.167	0.193	0.176	0.162	0.183	0.002	0.001	0.002	0.002	0.001	0.001
Nursery	1.609	1.226	5.312	1.558	1.217	5.083	0.093	0.098	1.000	0.086	0.047	0.085
Heart	0.264	0.204	0.264	0.295	0.204	0.267	0.006	0.010	0.003	0.014	0.010	0.004
PageBlocks	0.873	0.769	2.870	0.852	0.785	2.847	0.009	0.009	0.012	0.011	0.009	0.014
Dermatology	0.266	0.228	0.293	0.282	0.237	0.295	0.007	0.006	0.004	0.013	0.010	0.004
Glass	0.200	0.178	0.228	0.205	0.174	0.230	0.003	0.002	0.002	0.002	0.002	0.002
Zoo	0.128	0.123	0.134	0.139	0.123	0.161	0.001	0.001	0.001	0.001	0.001	0.001
Ecoli	0.237	0.201	0.229	0.248	0.204	0.255	0.002	0.001	0.002	0.002	0.003	0.002
Led	0.568	0.532	0.722	0.586	0.544	0.745	0.013	0.012	0.015	0.019	0.017	0.020
PenDigits	4.652	2.289	9.253	3.550	1.991	7.644	0.091	0.090	0.099	0.469	0.090	0.084
Soybean	0.533	0.485	0.614	0.532	0.460	0.593	0.006	0.006	0.008	0.006	0.005	0.011
ChessKRvK	24.583	2.946	40.116	31.808	2.876	43.766	0.365	0.385	0.422	0.361	0.243	0.345
LetterRecog	15.037	6.004	34.139	11.083	4.191	30.764	0.282	0.242	0.315	0.225	0.294	0.328
Mean	3.670	1.183	7.013	3.800	**1.018**	6.854	0.091	0.089	0.139	0.090	**0.083**	0.094

TABLE 3.4: Average Accuracy and AUC values obtained using Naive Bayes classifiers and Bagging of Naive Bayes classifiers at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting with respect to the Single Path Strategy

Data set	K-means&N		DS&N		HC&N		K-means&NB		DS&NB		HC&NB	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	68.42	0.69	**74.30**	**0.74**	43.12	0.43	67.82	0.68	74.28	**0.74**	57.70	0.58
Wine	92.54	0.93	**94.49**	**0.95**	28.31	0.34	90.19	0.90	94.31	**0.95**	44.45	0.44
Nursery	**91.92**	**0.56**	90.12	0.44	31.80	0.17	88.42	0.52	90.07	0.44	36.56	0.18
Heart	53.76	0.37	**57.70**	**0.41**	19.39	0.20	52.53	0.36	54.66	0.40	14.21	0.16
PageBlocks	**92.44**	**0.44**	91.96	0.34	1.68	0.22	89.77	0.20	91.83	0.29	73.44	0.23
Dermatology	78.64	0.75	**79.80**	**0.79**	10.60	0.17	78.07	0.75	77.67	0.76	21.55	0.19
Glass	62.04	0.41	**63.94**	**0.43**	15.01	0.12	57.28	0.36	59.26	0.40	20.40	0.12
Zoo	**95.09**	**0.60**	93.18	0.59	12.00	0.13	92.09	0.56	92.18	0.58	17.91	0.15
Ecoli	79.98	0.35	**82.31**	**0.36**	15.17	0.13	77.51	0.28	77.60	0.29	27.48	0.11
Led	61.22	0.61	60.16	0.60	15.16	0.15	64.53	0.64	**65.50**	**0.66**	12.50	0.13
PenDigits	82.68	0.82	68.56	0.68	10.69	0.10	**85.03**	**0.85**	65.64	0.65	11.05	0.11
Soybean	79.00	**0.84**	**79.55**	0.81	7.30	0.07	55.48	0.58	74.03	0.75	11.24	0.09
ChessKRvK	**46.28**	**0.42**	35.18	0.27	5.69	0.06	39.87	0.35	29.01	0.19	12.05	0.06
LetterRecog	**41.40**	**0.41**	39.16	0.39	4.85	0.05	34.74	0.35	35.05	0.35	3.80	0.04
Mean	**73.24**	**0.59**	72.17	0.56	15.77	0.17	69.52	0.53	70.08	0.53	26.02	0.19

TABLE 3.5: Run time results (in seconds) obtained using Naive Bayes classifiers and Bagging of Naive Bayes classifiers at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	Generation Time						Single Path Classification Time					
	K-means& N	DS& N	HC& N	K-means& NB	DS& NB	HC& NB	K-means& N	DS& N	HC& N	K-means& NB	DS& NB	HC& NB
Waveform	0.839	0.730	1.880	1.124	0.747	1.948	0.011	0.003	0.013	0.010	0.008	0.015
Wine	0.213	0.167	0.180	0.176	0.170	0.160	0.001	0.000	0.001	0.001	0.001	0.001
Nursery	1.197	1.048	4.452	1.420	1.146	4.591	0.017	0.008	0.017	0.012	0.010	0.019
Heart	0.295	0.229	0.226	0.437	0.873	0.205	0.001	0.001	0.001	0.002	0.012	0.001
PageBlocks	0.763	0.741	2.598	1.023	0.883	2.789	0.007	0.005	0.010	0.010	0.012	0.012
Dermatology	0.241	0.218	0.248	0.323	0.228	0.227	0.001	0.000	0.001	0.001	0.001	0.001
Glass	0.209	0.197	0.192	0.328	0.184	0.174	0.001	0.001	0.001	0.002	0.001	0.001
Zoo	0.146	0.127	0.131	0.158	0.142	0.115	0.001	0.001	0.001	0.001	0.002	0.001
Ecoli	0.215	0.197	0.211	0.345	0.197	0.193	0.001	0.001	0.001	0.003	0.001	0.001
Led	0.558	0.530	0.692	0.579	0.527	0.663	0.004	0.003	0.006	0.012	0.007	0.008
PenDigits	1.359	1.138	5.138	1.886	1.217	5.175	0.011	0.014	0.015	0.016	0.015	0.019
Soybean	0.458	0.362	0.460	0.649	0.494	0.438	0.002	0.001	0.003	0.004	0.003	0.005
ChessKRvK	2.021	1.555	17.740	4.468	1.925	18.288	0.046	0.022	0.060	0.063	0.024	0.065
LetterRecog	1.998	1.481	16.636	4.078	1.745	17.207	0.027	0.018	0.039	0.038	0.035	0.043
Mean	0.751	**0.623**	3.627	1.214	0.748	3.727	0.009	**0.006**	0.012	0.013	0.009	0.014

The results obtained with respect to the run-time experiments for Naive Bayes classifiers and Bagging of Naive Bayes classifiers at nodes coupled with the three considered data distribution techniques are presented in Table 3.5. As before the table shows the generation and classification times for each. Again, as in the case of the Decision tree hierarchies, the lowest generation and classification times were obtained when using the data splitting technique, followed by k-means and hierarchical clustering respectively. Regarding the classification style used at each tree node; using stand-alone Naive classifiers resulted in a lower generation and classification times than when using Bagging of Naive Bayes classifiers. From the table it can be observed that the lowest generation and classification times were obtained when using stand-alone Naive Bayes and data splitting to generate the binary tree hierarchical classification model.

With respect to using CARM to generate the base classifiers in the desired Binary Tree hierarchical classification model, a confidence threshold of 40% ($\tau = 40$ %) and a support threshold of 1% ($s = 1\%$) were used. A range of alternative threshold values were considered, not reported here, and it was found that ($\tau = 40$ %, and $s = 1\%$) produced the best performance with respect to the considered evaluation datasets. Note here that a low support value was used so as not miss any frequent item sets, that might result in a valid classification rule.

The results obtained when using CARM classifiers at each tree node, when using both stand-alone CARM and Bagging of CARM, coupled with the three alternative data distribution techniques are presented in Table 3.6. From the table it can be clearly observed, as in the case of the Decision Tree and Naive Bayes hierarchies, that using single (stand-alone) CARM classifiers at each tree node outperformed using Bagging of CARM classifiers at each node, regardless of the adopted data distribution technique. With respect to comparing the three considered data distribution techniques, it can be noted that in this case best results were generated using k-means, followed by data splitting and hierarchical clustering respectively. However, the statistical tests results indicated that there was no statistically significant difference between the three considered data distribution techniques with respect to CARM classifier.

The associated results obtained for the run-time experiments with respect to CARM classifiers and Bagging of CARM classifiers at nodes coupled with the three considered data distribution techniques: are presented in Table 3.7. As before the table gives the generation and classification times for each. The results corroborate the results obtained when using the Decision tree and Naive Bayes hierarchies presented above; the lowest generation and classification times were obtained when using data splitting, followed by k-means and hierarchical clustering respectively. Regarding the classification style used at each tree node, using stand-alone CARM classifiers resulted in lower generation and classification times than when using Bagging of CARM classifiers. From the table it can be observed that the lowest generation and classification times were obtained when using stand-alone CARM and data splitting to generate the binary tree hierarchical classification model.

TABLE 3.6: Average Accuracy and AUC values obtained using CARM classification and Bagging of CARM classification at nodes coupled with the three alternative data distribution techniques: (i) k-means, (ii) divisive hierarchical clustering and (iii) data splitting with respect to the Single Path Strategy

Data set	K-means&CARM		DS&CARM		HC&CARM		K-means&CARMB		DS&CARMB		HC&CARMB	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	58.92	0.59	57.02	0.58	**61.62**	**0.61**	40.50	0.40	57.80	0.58	42.84	0.42
Wine	78.99	0.81	78.59	0.76	**84.28**	**0.86**	65.62	0.66	70.33	0.68	70.54	0.70
Nursery	84.41	**0.43**	**86.81**	**0.43**	53.07	0.27	82.11	0.42	80.10	0.39	43.80	0.22
Heart	**53.76**	**0.24**	52.39	0.20	53.35	0.23	51.97	0.22	52.45	0.22	52.04	0.21
PageBlocks	**90.79**	0.24	89.77	0.20	73.73	**0.25**	89.77	0.20	89.77	0.20	72.84	0.20
Dermatology	**74.92**	0.64	60.28	0.46	71.39	**0.66**	62.85	0.47	57.87	0.40	49.58	0.43
Glass	48.46	0.29	**61.56**	**0.31**	52.19	0.30	48.21	0.28	59.49	0.28	46.95	0.26
Zoo	**88.00**	**0.52**	85.00	0.49	85.00	0.49	76.09	0.39	77.09	0.40	69.09	0.36
Ecoli	65.15	**0.28**	66.57	0.20	52.17	0.21	67.48	0.24	67.23	0.19	**68.79**	0.26
Led	**54.78**	**0.55**	45.53	0.45	35.16	0.35	28.63	0.28	25.06	0.24	15.78	0.15
PenDigits	**61.12**	**0.61**	42.68	0.42	50.74	0.51	54.68	0.54	55.96	0.56	35.92	0.36
Soybean	79.01	0.76	**88.97**	**0.89**	57.36	0.52	62.50	0.46	76.13	0.76	46.98	0.38
ChessKRvK	**32.99**	**0.18**	28.13	0.14	22.50	0.11	23.67	0.11	10.35	0.08	18.83	0.10
LetterRecog	29.58	0.30	**33.12**	**0.33**	21.54	0.22	12.87	0.13	20.01	0.20	13.00	0.13
Mean	**64.35**	**0.46**	62.60	0.42	55.29	0.40	54.78	0.34	57.12	0.37	46.21	0.30

TABLE 3.7: Run time results (in seconds) obtained using CARM classification and Bagging of CARM classification at nodes coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	Generation Time						Single Path Classification Time					
	K-means CARM	DS& CARM	HC& CARM	K-means& CARMB	DS& CARMB	HC& CARMB	K-means& CARM	DS& CARM	HC& CARM	K-means& CARMB	DS& CARMB	HC& CARMB
Waveform	12.296	1.420	12.647	31.626	1.927	30.558	0.009	0.009	0.009	0.013	0.043	0.014
Wine	0.473	0.460	0.996	1.376	0.957	2.182	0.001	0.001	0.001	0.001	0.001	0.001
Nursery	1.513	1.186	5.233	2.294	1.347	5.934	0.011	0.006	0.011	0.014	0.009	0.015
Heart	5.115	0.724	3.982	7.428	1.767	5.883	0.002	0.001	0.001	0.001	0.001	0.002
PageBlocks	0.978	0.912	2.964	1.127	1.087	3.266	0.003	0.002	0.006	0.003	0.005	0.007
Dermatology	2.120	0.801	2.918	3.452	2.018	4.709	0.001	0.001	0.001	0.002	0.003	0.002
Glass	0.406	0.298	0.456	0.581	0.437	0.608	0.001	0.001	0.001	0.001	0.001	0.002
Zoo	1.270	0.925	1.468	2.513	2.253	3.801	0.000	0.001	0.000	0.001	0.000	0.001
Ecoli	0.404	0.332	0.393	0.374	0.321	0.425	0.001	0.001	0.000	0.002	0.001	0.002
Led	0.665	0.593	0.829	0.791	0.711	0.918	0.002	0.002	0.008	0.005	0.006	0.007
PenDigits	37.004	4.913	57.562	111.709	7.267	165.690	0.015	0.013	0.017	0.018	0.015	0.016
Soybean	6.182	4.804	11.658	10.525	8.275	18.811	0.004	0.002	0.003	0.006	0.006	0.006
ChessKRvK	4.068	1.510	19.765	6.144	1.929	21.755	0.049	0.026	0.045	0.054	0.023	0.058
LetterRecog	203.600	11.160	340.029	313.510	17.222	365.035	0.024	0.025	0.061	0.049	0.026	0.089
Mean	19.721	**2.146**	32.921	35.246	3.394	44.970	0.009	**0.007**	0.012	0.012	0.010	0.016

Table 3.8 presents the results obtained using the Single Path strategy coupled with the three alternative classification algorithms for generating the node classifiers in the Binary Tree: Decision Tree, Naive Bayes, and CARM, with respect to the three considered data distribution techniques. Because the foregoing discussion established that it is not effective to use Bagging at each binary tree node, Bagging results are not presented in the table. From the table it can be seen that the best results, with respect to the majority of the datasets considered, were obtained when using Naive Bayes classifiers at the tree nodes. In addition the statistical tests results indicated that Naive Bayes classifier significantly outperformed Decision Tree and CARM classifiers.

As noted earlier with respect to the usage of Naive Bayes classifiers at tree nodes, the results produced when using k-means and data splitting are similar, and clearly outperform the hierarchical clustering results. However, the best overall results, in terms of average AUC, was obtained when Naive classifiers were used at each tree node coupled with the k-means technique for distributing data between nodes within the hierarchy.

Regarding the efficiency comparison between the three alternative classification algorithms for generating the node classifiers in the Binary Tree, from Tables 3.3, 3.5, and 3.7 it can be observed that the lowest generation and classification times were obtained when using Naive Bayes classifiers at the tree nodes.

From the above discussion, we can conclude that the choice of: (i) the base classifier and (ii) the data distribution technique, to generate the proposed Binary Tree hierarchical classification model, can significantly affect the classification accuracy of the resulting ensemble model. The most effective and efficient classifier was found to be the Naive Bayes classifier. Regarding the data distribution technique it was found that k-means and data splitting results are similar (not statistically significant), and significantly outperformed the hierarchical clustering results.

3.4.2 Multiple Path Experiments and Results

This section presents the results obtained using the Binary Tree hierarchical classification model coupled with the Multiple Path strategy. As noted earlier, the Multiple Path strategy was realised using the Naive Bayes and CARM classification models because these featured probability and confidence values respectively that could be used to determine whether single or multiple paths should be followed. Consequently, this section is divided into three sub-sections as follows: (i) Sub-section 3.4.2 discuss the conducted experiments, and the obtained results, when following multiple paths using Naive Bayes classifiers with respect to the different considered distribution techniques, and the three class label selection mechanisms (BIP, NAP and Voting), (ii) Sub-section 3.4.2 presents the conducted experiments, and the obtained results, when following multiple paths using CARM as the base classifier with respect to the different considered distribution techniques, and the three class label selection mechanisms (BIC, NAC and Voting); and (iii) Sub-section 3.4.2 presents a comparison between the two.

TABLE 3.8: Average Accuracy and AUC values obtained using the Single Path strategy coupled with the three alternative classification algorithms for generating the node classifiers in the Binary Tree: (i) Decision Tree, (ii) Naive Bayes and (iii) CARM, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means						Data splitting						*divisive* hierarchical clustering					
	DT		Naive		CARM		DT		Naive		CARM		DT		Naive		CARM	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	59.68	0.60	68.42	0.69	58.92	0.59	58.56	0.59	**74.30**	**0.74**	57.02	0.58	58.98	0.59	43.12	0.43	61.62	0.61
Wine	68.75	0.67	92.54	0.93	78.99	0.81	65.22	0.64	**94.49**	**0.95**	78.59	0.76	70.54	0.68	28.31	0.34	84.28	0.86
Nursery	35.69	0.19	**91.92**	**0.56**	84.41	0.43	33.51	0.27	90.12	0.44	86.81	0.43	59.70	0.32	31.80	0.17	53.07	0.27
Heart	47.97	0.31	53.76	0.37	53.76	0.24	49.89	0.32	**57.70**	**0.41**	52.39	0.20	47.48	0.26	19.39	0.20	53.35	0.23
PageBlocks	**92.56**	**0.49**	92.44	0.44	90.79	0.24	92.40	0.48	91.96	0.34	89.77	0.20	74.28	0.36	1.68	0.22	73.73	0.25
Dermatology	58.90	0.55	78.64	0.75	74.92	0.64	61.37	0.60	**79.80**	**0.79**	60.28	0.46	60.25	0.54	10.60	0.17	71.39	0.66
Glass	60.44	0.39	62.04	0.41	48.46	0.29	60.91	0.38	**63.94**	**0.43**	61.56	0.31	56.95	0.34	15.01	0.12	52.19	0.30
Zoo	85.00	0.50	**95.09**	**0.60**	88.00	0.52	85.00	0.50	93.18	0.59	85.00	0.49	82.09	0.51	12.00	0.13	85.00	0.49
Ecoli	76.90	0.33	79.98	0.35	65.15	0.28	78.72	0.35	**82.31**	**0.36**	66.57	0.20	56.29	0.28	15.17	0.13	52.17	0.21
Led	**75.00**	**0.75**	61.22	0.61	54.78	0.55	**75.00**	**0.75**	60.16	0.60	45.53	0.45	59.94	0.59	15.16	0.15	35.16	0.35
PenDigits	67.30	0.67	**82.68**	**0.82**	61.12	0.61	66.78	0.67	68.56	0.68	42.68	0.42	66.30	0.66	10.69	0.10	50.74	0.51
Soybean	64.62	0.65	79.00	0.84	79.01	0.76	63.02	0.56	79.55	0.81	**88.97**	**0.89**	52.67	0.47	7.30	0.07	57.36	0.52
ChessKRvK	25.97	0.21	**46.28**	**0.42**	32.99	0.18	27.41	0.25	35.18	0.27	28.13	0.14	24.67	0.18	5.69	0.06	22.50	0.11
LetterRecog	**42.99**	**0.43**	41.40	0.41	29.58	0.30	42.76	**0.43**	39.16	0.39	33.12	0.33	38.59	0.39	4.85	0.05	21.54	0.22
Mean	61.56	0.48	**73.24**	**0.59**	64.35	0.46	61.47	0.49	72.17	0.56	62.60	0.42	57.77	0.44	15.77	0.17	55.29	0.40

Because the foregoing section (Section 3.4.1) established that it is not effective nor efficient to use Bagging classification at each tree node, the results presented in this section have all been generated using "stand-alone" classification at nodes.

Using Naive Bayesian Probability Values for Following Multiple Paths Within the Binary Tree Hierarchical Classification Model

In this section the conducted experiments and the obtained results when following multiple paths utilising Naive Bayes probability values, with respect to the different considered data distribution techniques, are presented. This section also presents a comparison between the three different proposed mechanisms for arriving at a final classification result: (i) BIP, (ii) NAP and (iii) Voting. Recall that the objective here was to identify the most effective mechanism for selecting the final class label for a given, previously unseen, example.

Using the BIP class label mechanism experiments using a range of alternative σ values were conducted that demonstrated that $\sigma = 0.1 \times 10^{-1}$, $\sigma = 0.1 \times 10^{-6}$ and $\sigma = 0.0$ produced the best performance with respect to k-means, data splitting and *divisive* hierarchical clustering respectively (for completeness these results are included in Appendix D). With respect to the NAP class label selection mechanism experiments using a range of alternative σ values demonstrated that $\sigma = 0.1 \times 10^{-2}$, $\sigma = 0.1 \times 10^{-6}$ and $\sigma = 0.0$ produced the best performance with respect to k-means, data splitting and *divisive* hierarchical clustering respectively (See Appendix D). Regarding the Voting mechanism $\sigma = 0.1 \times 10^{-2}$, $\sigma = 0.1 \times 10^{-2}$ and $\sigma = 0.0$ produced the best performance with respect to k-means, data splitting and *divisive* hierarchical clustering respectively (for completeness these results are included in Appendix D). Note here that σ is a Naive Bayesian probability value, $0 \leq \sigma < 1$, consequently low values are expected.

A comparison between the three alternative class label selection mechanisms: BIP, NAP and Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting, is presented in Table 3.9 (best AUC results with respect to each data distribution technique is highlighted in bold font). According to the table, the best average (mean) AUC was obtained when using the NAP mechanism and any of the three considered data distribution mechanisms. From the table it can be observed that the results of BIP and NAP mechanisms are similar, regardless of the data distribution technique; slightly better results are perhaps obtained using the NAP mechanism. In addition, NAP and BIP outperformed the Voting mechanism, regardless of the data distribution technique. The reason why the Voting mechanism sometimes produced worse results than the NAP or BIP mechanisms is that the Voting mechanism can be significantly affected by votes associated with inaccurate paths whereas the NAP or BIP mechanisms assign a specific weights to each candidate class thus avoiding the problem of counting votes from inaccurate paths. While the reason why the BIP mechanism sometimes produced worse results than the NAP mechanism is that the BIP mechanism depends only on the classification result

TABLE 3.9: Average Accuracy and AUC values obtained using the Multiple Path strategy coupled with the three alternative class label selection mechanisms: (i) BIP, (ii) NAP and (iii) Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means						Data Splitting						*divisive* hierarchical clustering					
	BIP		NAP		Voting		BIP		NAP		Voting		BIP		NAP		Voting	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	68.42	**0.69**	68.42	**0.69**	68.42	**0.69**	74.98	0.75	76.44	**0.76**	74.30	0.74	53.36	0.53	53.62	0.53	55.90	**0.56**
Wine	92.54	**0.93**	92.54	**0.93**	92.54	**0.93**	95.67	**0.96**	96.26	**0.96**	94.49	0.95	54.82	0.57	56.59	**0.58**	46.80	0.48
Nursery	91.90	0.56	91.88	**0.57**	91.91	0.56	87.41	**0.58**	89.09	**0.58**	90.12	0.44	46.49	**0.30**	46.37	**0.30**	42.36	0.28
Heart	53.42	0.37	53.01	**0.38**	53.49	0.36	53.08	0.38	53.77	0.36	57.70	**0.41**	23.39	**0.22**	24.43	**0.22**	11.18	0.17
PageBlocks	92.34	0.44	92.09	**0.45**	91.56	0.31	90.10	0.45	91.27	**0.48**	91.96	0.34	82.38	0.34	82.38	**0.36**	82.05	0.26
Dermatology	77.50	0.76	74.53	0.71	79.62	**0.78**	82.94	0.82	84.60	**0.84**	79.80	0.79	41.95	0.33	43.28	**0.34**	41.22	0.30
Glass	59.73	0.40	57.35	0.39	62.51	**0.41**	50.99	0.49	55.28	**0.51**	63.94	0.43	48.06	**0.34**	46.23	**0.34**	46.48	0.16
Zoo	95.09	**0.60**	95.09	**0.60**	95.09	**0.60**	91.27	0.58	92.18	0.58	93.18	**0.59**	80.36	0.51	84.18	**0.52**	26.00	0.13
Ecoli	76.75	**0.32**	74.62	**0.32**	78.07	0.31	68.65	0.34	64.15	0.27	82.31	**0.36**	43.68	0.27	44.59	**0.28**	55.92	0.23
Led	52.69	0.53	70.72	**0.71**	48.13	0.48	60.41	0.60	61.13	**0.61**	47.38	0.48	44.72	**0.44**	44.16	**0.44**	25.66	0.26
PenDigits	82.66	**0.82**	82.47	**0.82**	82.66	**0.82**	83.30	**0.83**	81.18	0.81	68.56	0.68	59.58	0.60	60.80	**0.61**	23.11	0.24
Soybean	79.00	**0.84**	78.82	**0.84**	79.00	**0.84**	75.45	0.73	83.71	**0.83**	79.55	0.81	60.67	0.69	64.23	**0.71**	24.93	0.14
ChessKRvK	45.26	**0.42**	42.79	0.39	45.11	0.41	28.58	0.35	33.88	**0.37**	35.18	0.27	19.92	**0.22**	18.59	0.21	14.61	0.06
LetterRecog	41.32	**0.41**	41.20	**0.41**	41.30	**0.41**	54.59	**0.54**	53.44	0.53	39.16	0.39	28.89	0.29	30.41	**0.30**	9.02	0.09
Mean	72.04	0.58	72.54	**0.59**	72.10	0.57	71.24	0.60	72.60	**0.61**	71.26	0.55	49.16	0.40	49.99	**0.41**	36.09	0.24

from only a single classifier (last classifier in the path), while the NAP mechanism considers all the classifiers along the followed path. With respect to the statistical significance of these results, it was found that the NAP mechanism statistically outperformed the Voting mechanism for arriving at a final classification decision, while no statistically significant difference was detected in the operation of the BIP and NAP mechanisms with respect to data splitting and hierarchical clustering techniques. However, no statistically significant difference was found in performance between the three considered mechanisms with respect to k-means data distribution technique. The reason for this is that the successive mis-classification issue occurred more readily with data splitting and hierarchical clustering than in the case of k-means when using the single path classification strategy, consequently the differences in performance between the three considered mechanisms (NAP, BIP and Voting) were more noticeable with respect to data splitting and hierarchical clustering. The NAP mechanism was therefore considered to be the most appropriate mechanism for selecting the final resulting class label. NAP was thus the mechanism adopted with respect to the remaining experiments reported in this chapter.

From Table 3.9 it can also be observed that the best overall results, with respect to the Multiple Path strategy, were obtained when using the data splitting technique for data distribution. However, no statistical difference between k-means and data splitting technique was detected according to the conducted statistical tests.

The results obtained for the run-time experiments with respect to the Multiple Path strategy coupled with Naive Bayes classification and the three considered data distribution techniques, are presented in Table 3.10. The table gives both the generation and classification times for each. Again, and as in the case of the Single Path strategy, the lowest generation and classification times were obtained when using the data splitting technique, followed by k-means and hierarchical clustering respectively.

Using CARM Confidence Values for Following Multiple Paths Within the Binary Tree Hierarchical Classification Model

In this section the experimental results obtained when following multiple paths within the Binary Tree structure utilising the confidence values generated using CARM classifiers at each tree node are presented. This section also presents a comparison between the three different proposed mechanisms for arriving at a final classification result, BIC, NAC and Voting class label selection, in the context of CARM classification and with respect to the objective of identifying the most effective mechanism.

Experiments were conducted using BIC, NAC and Voting to identify the most appropriate value of σ in each case. Some detail concerning these experiments are presented in Appendix D. Using BIC, NAC and Voting mechanisms, it was found that $\sigma = 70$ produced the best performance regardless the adopted data distribution technique. Although, as a general rule $\sigma = 70$ had been found to produce the best performance for

TABLE 3.10: Run time results (in seconds) obtained using the Multiple Path strategy coupled with NAP with respect to Naive Bayes classification and the three considered data distribution techniques: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering

Data set	Multiple Path Classification Time		
	K-means&N	DS&N	HC&N
Waveform	0.285	0.226	0.247
Wine	0.009	0.009	0.009
Nursery	0.659	0.576	0.609
Heart	0.045	0.020	0.029
PageBlocks	0.254	0.266	0.281
Dermatology	0.028	0.021	0.026
Glass	0.015	0.011	0.015
Zoo	0.018	0.017	0.018
Ecoli	0.031	0.024	0.027
Led	0.171	0.153	0.189
PenDigits	0.732	0.507	0.554
Soybean	0.038	0.047	0.033
ChessKRvK	1.981	1.254	2.338
LetterRecog	1.176	0.909	1.126
Mean	0.389	**0.289**	0.393

most of the datasets considered, a specific best value for σ could be identified for each dataset.

A comparison between the three alternative class label selection mechanisms: BIC, NAC and Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting is presented in Table 3.11 (best AUC results with respect to each data distribution technique highlighted in bold font). From the table it can be observed that the results obtained using the three mechanisms, BIC, NAC and Voting, are very similar (not statistically significant). From Table 3.11 it can also be observed that the best overall results, with respect to the Multiple Path strategy, were obtained when using k-means technique for data distribution. The conducted statistical tests indicated that k-means distribution was more effective than data splitting. While the performance difference between k-means and hierarchical clustering was found to not be statistically different. Similarly no statistically significant performance difference was found between the data splitting and hierarchical clustering distribution techniques. The reason for this is that an issue with using confidence values, generated when using CARM, to determine whether one or two branches emanating from a node should be followed, was that it was not always possible to identify both branch confidence values for a given node. Consequently the unknown confidence values affected the results of Multiple Path strategy. This issue would affect the outcome of any comparison of the different data distribution techniques coupled with CARM classification and the Multiple Path strategy.

TABLE 3.11: Average Accuracy and AUC values obtained using the Multiple Path strategy coupled with the two alternative class label selection mechanisms: (i) BIC, (ii) NAC and (iii) Voting with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means						Data Splitting						*divisive* hierarchical clustering					
	BIC		NAC		Voting		BIC		NAC		Voting		BIC		NAC		Voting	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	59.02	**0.59**	59.02	**0.59**	58.90	**0.59**	57.12	**0.58**	57.44	**0.58**	57.02	**0.58**	61.58	**0.61**	61.58	**0.61**	61.54	**0.61**
Wine	78.99	**0.81**	78.99	**0.81**	78.79	**0.81**	78.59	**0.76**	78.59	**0.76**	78.59	**0.76**	84.28	**0.86**	84.28	**0.86**	84.28	**0.86**
Nursery	82.53	**0.42**	80.28	0.40	82.79	**0.42**	86.81	**0.43**	86.81	**0.43**	86.81	**0.43**	53.12	**0.27**	53.06	**0.27**	53.06	**0.27**
Heart	53.76	**0.24**	53.76	**0.24**	53.76	**0.24**	52.39	**0.20**	52.39	**0.20**	52.39	**0.20**	53.35	**0.23**	53.35	**0.23**	53.35	**0.23**
PageBlocks	90.79	0.24	91.17	**0.35**	90.79	0.24	89.77	**0.20**	89.77	**0.20**	89.77	**0.20**	73.88	0.23	73.75	**0.26**	73.88	0.23
Dermatology	75.73	0.65	77.34	**0.68**	74.64	0.64	60.28	**0.46**	60.28	**0.46**	60.28	**0.46**	71.16	**0.66**	71.16	**0.66**	71.16	**0.66**
Glass	48.46	**0.29**	48.93	**0.29**	46.48	0.26	61.56	**0.31**	61.56	**0.31**	58.20	0.20	51.24	**0.30**	51.71	**0.30**	50.11	0.27
Zoo	88.00	**0.52**	88.00	**0.52**	88.00	**0.52**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**
Ecoli	64.55	0.28	64.85	0.28	70.37	**0.29**	66.57	**0.20**	66.57	**0.20**	66.57	**0.20**	51.86	0.20	52.17	**0.21**	51.86	0.20
Led	52.16	**0.52**	52.13	**0.52**	48.56	0.49	41.38	**0.41**	41.53	**0.41**	37.53	0.37	35.38	0.35	37.94	0.38	40.41	**0.40**
PenDigits	61.12	0.61	60.86	0.61	66.42	**0.66**	43.89	**0.43**	40.12	0.40	40.45	0.40	50.41	0.50	50.43	0.50	51.26	**0.51**
Soybean	79.01	**0.76**	79.01	**0.76**	79.01	**0.76**	88.43	**0.89**	88.43	**0.89**	88.43	**0.89**	56.82	**0.52**	56.83	**0.52**	56.64	**0.52**
ChessKRvK	32.16	**0.17**	31.06	**0.17**	32.13	**0.17**	26.58	**0.13**	27.28	**0.13**	24.66	**0.13**	22.48	**0.11**	22.20	**0.11**	22.80	**0.11**
LetterRecog	29.17	**0.29**	27.87	0.28	28.35	0.28	29.13	**0.29**	27.39	0.27	29.50	**0.29**	21.67	**0.22**	21.57	0.22	21.73	**0.22**
Mean	63.96	**0.46**	63.81	**0.46**	64.21	**0.46**	61.96	**0.41**	61.65	**0.41**	61.09	0.40	55.16	**0.40**	55.36	**0.40**	55.51	**0.40**

TABLE 3.12: Run time results (in seconds) obtained using the Multiple Path strategy coupled with NAP with respect to CARM classifiers and the three considered data distribution techniques: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering

Data set	Multiple Path Classification Time		
	K-means&CARM	DS&CARM	HC&CARM
Waveform	0.272	0.234	0.275
Wine	0.008	0.009	0.009
Nursery	0.577	0.599	0.725
Heart	0.017	0.019	0.023
PageBlocks	0.259	0.255	0.247
Dermatology	0.025	0.017	0.022
Glass	0.017	0.015	0.010
Zoo	0.011	0.006	0.005
Ecoli	0.019	0.024	0.017
Led	0.161	0.173	0.163
PenDigits	0.678	0.505	0.734
Soybean	0.048	0.037	0.038
ChessKRvK	1.343	1.254	1.404
LetterRecog	2.201	0.898	2.412
Mean	0.403	**0.289**	0.435

The results obtained for the run-time experiments with respect to the Multiple Path strategy using CARM classifiers coupled with the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting, are presented in Table 3.12. The table presents the generation and classification times for each. Again, and as in the case of the Single Path strategy, the lowest generation and classification times were obtained from using the data splitting technique, followed by k-means and hierarchical clustering respectively.

Comparison Between Using Probability Values and Confidence Values for Following Multiple Paths Within the Binary Tree Hierarchy

In this sub-section a comparison between using Naive Bayesian probability values and CARM confidence values, for following multiple paths within the Binary Tree hierarchical classification model, is presented. Recall that the objective of the comparison was to determine the most effective classifier to be utilised with respect to the proposed Multiple Path strategy.

Table 3.13 present the results obtained with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting (the best AUC results with respect to each data distribution technique are highlighted in bold font). Note here that the presented results were obtained using the Multiple Path strategy coupled with NAP, because the forgoing sections established that

NAP tended to produce more accurate results than BIP or Voting class label selection with respect to Naive Bayes classification, and that there was no observed difference between the NAC, BIC and Voting class label selection with respect to CARM. From Table 3.13 it can be clearly observed that utilising Naive Bayesian probability values significantly outperforms utilising CARM confidence values regardless the adopted data distribution technique.

Again, as the case of the Single Path strategy, Naive Bayes classifiers are the best choice for generating the Binary Tree hierarchical classification model, compared to the usage of CARM classifiers.

3.4.3 Comparison Between Single Path and Multiple Path Strategies

The objective of the comparison between the Single Path and Multiple Path strategies was to determine whether following more than one path within the proposed Binary Tree hierarchical classification model could address the successive mis-classification issue noted earlier.

Commencing with a comparison of the Single and Multiple path strategies with respect to Naive Bayes classification. As a result of experiments conducted previously, and presented above, the Multiple Path strategy coupled with the NAP mechanism was adopted for this purpose. Table 3.14 presents the results obtained with respect to the three considered data distribution techniques (best AUC results with respect to each data distribution technique are highlighted in bold font). From the table it can be observed that, with respect to using k-means as the data distribution technique, for six of the datasets the same AUC value was obtained regardless of which strategy (Single or Multiple) was adopted. For four of the datasets the multiple path strategy produced the best AUC value, for another four datasets the Single Path strategy produced the best AUC results. Thus a no difference, multiple path and single path ratio of $6 : 4 : 4$. When using data splitting as the data distribution technique the ratio was $0 : 11 : 3$. When using hierarchical clustering as the data distribution technique the ratio was $0 : 14 : 0$, thus the Multiple Path strategy produced the best results in all cases. In other words, from the table, it can be observed that:

1. The Multiple Path strategy clearly outperformed the Single Path strategy in the context of using data splitting and hierarchical clustering as the data distribution techniques; while there was no noticeable improvement with respect to k-means. The reason for this is that the successive mis-classification issue occurred more readily with data splitting and hierarchical clustering than in the case of k-means when using the single path classification strategy (best results were obtained using k-means). The conducted statistical tests supported this observation. More specifically, there was a statistical performance difference between the two strategies with respect to data splitting and hierarchical clustering. While there was no significant difference between the two strategies with respect to k-means.

TABLE 3.13: Average Accuracy and AUC values obtained using Naive Bayes and CARM to generate Binary Tree hierarchical classification models coupled with the Multiple Path strategy (using NAP mechanism), with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means				Data splitting				*divisive* hierarchical clustering			
	Naive		CARM		Naive		CARM		Naive		CARM	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	68.42	**0.69**	59.02	0.59	76.44	**0.76**	57.44	0.58	53.62	0.53	61.58	**0.61**
Wine	92.54	**0.93**	78.99	0.81	96.26	**0.96**	78.59	0.76	56.59	0.58	84.28	**0.86**
Nursery	91.88	**0.57**	80.28	0.40	89.09	**0.58**	86.81	0.43	46.37	**0.30**	53.06	0.27
Heart	53.01	**0.38**	53.76	0.24	53.77	**0.36**	52.39	0.20	24.43	0.22	53.35	**0.23**
PageBlocks	92.09	**0.45**	91.17	0.35	91.27	**0.48**	89.77	0.20	82.38	**0.36**	73.75	0.26
Dermatology	74.53	**0.71**	77.34	0.68	84.60	**0.84**	60.28	0.46	43.28	0.34	71.16	**0.66**
Glass	57.35	**0.39**	48.93	0.29	55.28	**0.51**	61.56	0.31	46.23	**0.34**	51.71	0.30
Zoo	95.09	**0.60**	88.00	0.52	92.18	**0.58**	85.00	0.49	84.18	**0.52**	85.00	0.49
Ecoli	74.62	**0.32**	64.85	0.28	64.15	0.27	66.57	0.20	44.59	**0.28**	52.17	0.21
Led	70.72	**0.71**	52.13	0.52	61.13	**0.61**	41.53	0.41	44.16	**0.44**	37.94	0.38
PenDigits	82.47	**0.82**	60.86	0.61	81.18	**0.81**	40.12	0.40	60.80	**0.61**	50.43	0.50
Soybean	78.82	**0.84**	79.01	0.76	83.71	0.83	88.43	**0.89**	64.23	**0.71**	56.83	0.52
ChessKRvK	42.79	**0.39**	31.06	0.17	33.88	**0.37**	27.28	0.13	18.59	**0.21**	22.20	0.11
LetterRecog	41.20	**0.41**	27.87	0.28	53.44	**0.53**	27.39	0.27	30.41	**0.30**	21.57	0.22
Mean	72.54	**0.59**	63.81	0.46	72.60	**0.61**	61.65	0.41	49.99	**0.41**	55.36	0.40

TABLE 3.14: Average Accuracy and AUC values obtained using Naive Bayes to generate Binary Tree hierarchical classification models coupled with the Single and Multiple Path strategies, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means&Naive				Data splitting&Naive				Hierarchical clustering&Naive			
	Single Path		Multiple Path		Single Path		Multiple Path		Single Path		Multiple Path	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	68.42	**0.69**	68.42	**0.69**	74.30	0.74	76.44	**0.76**	43.12	0.43	53.62	**0.53**
Wine	92.54	**0.93**	92.54	**0.93**	94.49	0.95	96.26	**0.96**	28.31	0.34	56.59	**0.58**
Nursery	91.92	0.56	91.88	**0.57**	90.12	0.44	89.09	**0.58**	31.80	0.17	46.37	**0.30**
Heart	53.76	0.37	53.01	**0.38**	57.70	**0.41**	53.77	0.36	19.39	0.20	24.43	**0.22**
PageBlocks	92.44	0.44	92.09	**0.45**	91.96	0.34	91.27	**0.48**	1.68	0.22	82.38	**0.36**
Dermatology	78.64	**0.75**	74.53	0.71	79.80	0.79	84.60	**0.84**	10.60	0.17	43.28	**0.34**
Glass	62.04	**0.41**	57.35	0.39	63.94	0.43	55.28	**0.51**	15.01	0.12	46.23	**0.34**
Zoo	95.09	**0.60**	95.09	**0.60**	93.18	**0.59**	92.18	0.58	12.00	0.13	84.18	**0.52**
Ecoli	79.98	**0.35**	74.62	0.32	82.31	**0.36**	64.15	0.27	15.17	0.13	44.59	**0.28**
Led	61.22	0.61	70.72	**0.71**	60.16	0.60	61.13	**0.61**	15.16	0.15	44.16	**0.44**
PenDigits	82.68	**0.82**	82.47	**0.82**	68.56	0.68	81.18	**0.81**	10.69	0.10	60.80	**0.61**
Soybean	79.00	**0.84**	78.82	**0.84**	79.55	0.81	83.71	**0.83**	7.30	0.07	64.23	**0.71**
ChessKRvK	46.28	**0.42**	42.79	0.39	35.18	0.27	33.88	**0.37**	5.69	0.06	18.59	**0.21**
LetterRecog	41.40	**0.41**	41.20	**0.41**	39.16	0.39	53.44	**0.53**	4.85	0.05	30.41	**0.30**
Mean	73.24	**0.59**	72.54	**0.59**	72.17	0.56	72.60	**0.61**	15.77	0.17	49.99	**0.41**

2. The best overall results were obtained when following multiple paths within the hierarchies generated using data splitting, however, not significantly outperformed following multiple paths within the hierarchies generated using k-means. An issue with using clustering algorithms to distribute class labels between nodes within the hierarchy is that similar classes were grouped together early on in the process so that entire branches ended up dealing with very similar classes, ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built.

Regarding the use of CARM classifiers to generate the proposed Binary Tree model, Table 3.15 presents the average accuracy and AUC results obtained when adopting either a Single or a Multiple path strategy within a the Binary Tree hierarchy with respect to the three considered data distribution techniques (again, the best obtained AUC result, with respect to each data distribution technique, is highlighted in bold font). From the table it can be observed that, with respect to using k-means as the data distribution technique, for eight datasets the same AUC value was obtained regardless of which strategy (Single or Multiple) was adopted. For two datasets the Multiple Path strategy produced the best AUC value, and for four datasets the Single Path strategy produced the best AUC results. A no difference, multiple path and single path ratio of $8 : 2 : 4$. When using data splitting the ratio was $10 : 0 : 4$. Using hierarchical clustering the ratio was $11 : 2 : 1$. The reason for the weakness of the Multiple Path strategy, in context of CARM classifiers as mentioned in section 3.4.2, is that it was not always possible to identify both confidence values for both branches emanating from a given node. Consequently the unknown confidence values affected the results obtained using the Multiple Path strategy, thus again resulting in it being of less value than initially anticipated. This issue would affect the outcome of any statistical comparison as concluded in Section 3.4.2, consequently no further statistical tests were conducted.

Regarding run time efficiency, of course the classification time required when using the Single Path strategy is less than that required when using the multiple path strategy (see Tables 3.5, 3.10, 3.7 and 3.12).

3.4.4 Comparison Between The Binary Tree Hierarchical Classification Model and Conventional models

In this section a comparison between the proposed Binary Tree hierarchical classification model and conventional classification models is presented. In order to conduct a consistent comparison between the proposed Binary Tree hierarchical ensembles and existing conventional models the comparison was conducted using the same classifier generator in each case. Three set of experiments are reported on here: (i) comparison between the operation of a stand-alone Decision Tree classification, Bagging of decision trees and the decision tree based Binary Tree hierarchical model, (ii) comparison between the operation of stand-alone Naive Bayes classification, Bagging of Naive Bayes classifiers

TABLE 3.15: Average Accuracy and AUC values obtained using CARM to generate Binary Tree hierarchical classification models coupled with the Single and Multiple Path strategies, with respect to the three considered data distribution techniques: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting

Data set	k-means&CARM				Data splitting&CARM				Hierarchical clustering&CARM			
	Single Path		Multiple Path		Single Path		Multiple Path		Single Path		Multiple Path	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	58.92	**0.59**	59.02	**0.59**	57.02	**0.58**	57.44	**0.58**	61.62	**0.61**	61.58	**0.61**
Wine	78.99	**0.81**	78.99	**0.81**	78.59	**0.76**	78.59	**0.76**	84.28	**0.86**	84.28	**0.86**
Nursery	84.41	**0.43**	80.28	0.40	86.81	**0.43**	86.81	**0.43**	53.07	**0.27**	53.06	**0.27**
Heart	53.76	**0.24**	53.76	**0.24**	52.39	**0.20**	52.39	**0.20**	53.35	**0.23**	53.35	**0.23**
PageBlocks	90.79	0.24	91.17	**0.35**	89.77	**0.20**	89.77	**0.20**	73.73	0.25	73.75	**0.26**
Dermatology	74.92	0.64	77.34	**0.68**	60.28	**0.46**	60.28	**0.46**	71.39	**0.66**	71.16	**0.66**
Glass	48.46	**0.29**	48.93	**0.29**	61.56	**0.31**	61.56	**0.31**	52.19	**0.30**	51.71	**0.30**
Zoo	88.00	**0.52**	88.00	**0.52**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**	85.00	**0.49**
Ecoli	65.15	**0.28**	64.85	**0.28**	66.57	**0.20**	66.57	**0.20**	52.17	**0.21**	52.17	**0.21**
Led	54.78	**0.55**	52.13	0.52	45.53	**0.45**	41.53	0.41	35.16	0.35	37.94	**0.38**
PenDigits	61.12	**0.61**	60.86	**0.61**	42.68	**0.42**	40.12	0.40	50.74	**0.51**	50.43	0.50
Soybean	79.01	**0.76**	79.01	**0.76**	88.97	**0.89**	88.43	**0.89**	57.36	**0.52**	56.83	**0.52**
ChessKRvK	32.99	**0.18**	31.06	0.17	28.13	**0.14**	27.28	0.13	22.50	**0.11**	22.20	**0.11**
LetterRecog	29.58	**0.30**	27.87	0.28	33.12	**0.33**	27.39	0.27	21.54	**0.22**	21.57	**0.22**
Mean	64.35	**0.46**	63.81	**0.46**	62.60	**0.42**	61.65	0.41	55.29	**0.40**	55.36	**0.40**

and the Naive Bayes based Binary Tree hierarchical classification model; and (iii) comparison between the operation of a stand-alone CARM classifier, Bagging of CARM and the CARM based Binary Tree hierarchical model.

Starting with the comparison between "stand-alone" Decision Tree classification, Bagging of decision trees, and the proposed Binary Tree hierarchical classification model with decision tree classifiers at each node. Table 3.16 presents the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). From the table it can be observed that the Binary Tree hierarchies produce an improved classification accuracy with respect to seven of the fourteen datasets considered. In the remaining seven cases, the stand-alone Decision Tree classifier produced the best AUC result (although in one case the AUC result was the same as that produced using our hierarchical approach and in another case the same as that produced using Bagging). With respect to the statistical evaluation, no statistically significant difference was detected in the operation of the k-means&DT and the Decision Tree classification (both stand-alone and Bagging). Similarly, no statistically significant difference was detected in the operation of the DS&DT and the Decision Tree classification (both stand-alone and Bagging). Regarding HC&DT, no statistically significant difference was detected in the operation of the HC&DT and Bagging, however, stand-alone Decision Tree significantly outperformed HC&DT.

With respect to the comparison between "stand-alone" Naive Bayes classification, Bagging of Naive Bayes classifiers and Naive Bayes Binary Tree hierarchies Table 3.17 presents the results obtained in terms of average accuracy and average AUC (as before best AUC results highlighted in bold font). Note here that the results presented with respect to the Binary Tree hierarchies are the Multiple Path results. From the table it can be observed that the proposed Binary Tree hierarchical classification model improves classification accuracy with respect to five of the fourteen datasets considered, although for one datasets the same result was produced as when Naive Bayes classification was used in stand-alone mode. For another seven datasets the stand-alone Naive Bayes classifier produced the best result although for five datasets the same result was produced when Bagging was used. For two datasets Bagging produced the best results. With respect to the statistical evaluation, no statistically significant difference was detected in the operation of the k-means&N and the Naive Bayes classification (both stand-alone and Bagging). Similarly, no statistically significant difference was detected in the operation of the DS&N and the Naive Bayes classification (both stand-alone and Bagging). Regarding HC&DT, Naive Bayes classification (both stand-alone and Bagging) significantly outperformed HC&N.

Regarding the comparison between "stand-alone" CARM, Bagging of CARM, and the CARM based Binary Tree hierarchies, Table 3.18 presents the results obtained in terms of average accuracy and average AUC (best AUC results highlighted in bold font). From the table it can be observed that the Binary Tree hierarchies produced the best

TABLE 3.16: Average accuracy and AUC results obtained when using: (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) Binary Tree hierarchies with decision trees at nodes (K-means&DT, DS&DT, and HC&DT)

Data set	DT		Bagging		K-means&DT		DS&DT		HC&DT	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
WaveForm	53.72	0.54	53.36	0.53	**59.68**	**0.60**	58.56	0.59	58.98	0.59
Wine	**73.86**	**0.73**	69.91	0.69	68.75	0.67	65.22	0.64	70.54	0.68
Nursery	5.15	0.03	32.71	0.16	35.69	0.19	33.51	0.27	**59.70**	**0.32**
Heart	48.80	0.28	51.15	0.28	47.97	0.31	**49.89**	**0.32**	47.48	0.26
PageBlocks	92.55	**0.49**	92.23	0.47	**92.56**	**0.49**	92.40	0.48	74.28	0.36
Dermatology	57.53	0.57	43.95	0.39	58.90	0.55	**61.37**	**0.60**	60.25	0.54
Glass	**64.50**	**0.40**	62.27	0.36	60.44	0.39	60.91	0.38	56.95	0.34
Zoo	**89.00**	**0.53**	87.27	**0.53**	85.00	0.50	85.00	0.50	82.09	0.51
Ecoli	78.07	0.34	73.61	0.31	76.90	0.33	**78.72**	**0.35**	56.29	0.28
Led	74.72	0.74	74.06	0.74	**75.00**	**0.75**	**75.00**	**0.75**	59.94	0.59
PenDigits	**76.84**	**0.77**	72.64	0.72	67.30	0.67	66.78	0.67	66.30	0.66
Soybean	52.12	0.52	37.18	0.30	**64.62**	**0.65**	63.02	0.56	52.67	0.47
ChessKRVK	**40.46**	**0.29**	30.07	0.16	25.97	0.21	27.41	0.25	24.67	0.18
LetRecog	**56.05**	**0.56**	41.82	0.42	42.99	0.43	42.76	0.43	38.59	0.39
Mean	**61.67**	**0.49**	58.73	0.43	61.56	0.48	61.47	**0.49**	57.77	0.44

TABLE 3.17: Average accuracy and AUC results obtained when using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes and (iii) Binary Tree hierarchies with Naive Bayes classifiers at nodes (K-means&N, DS&N, and HC&N)

Data set	Naive Bayes		Bagging		K-means&N		DS&N		HC&N	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	77.04	**0.77**	77.06	**0.77**	68.42	0.69	76.44	0.76	53.62	0.53
Wine	95.67	**0.96**	93.72	0.94	92.54	0.93	96.26	**0.96**	56.59	0.58
Nursery	90.22	0.45	89.96	0.46	91.88	0.57	89.09	**0.58**	46.37	0.30
Heart	54.60	0.34	51.28	0.30	53.01	**0.38**	53.77	0.36	24.43	0.22
PageBlocks	92.69	**0.52**	92.62	**0.52**	92.09	0.45	91.27	0.48	82.38	0.36
Dermatology	86.66	**0.85**	81.00	0.81	74.53	0.71	84.60	0.84	43.28	0.34
Glass	67.83	0.49	55.28	0.46	57.35	0.39	55.28	**0.51**	46.23	0.34
Zoo	92.27	0.59	94.27	**0.62**	95.09	0.60	92.18	0.58	84.18	0.52
Ecoli	81.70	0.38	82.56	**0.39**	74.62	0.32	64.15	0.27	44.59	0.28
Led	75.59	**0.76**	75.50	**0.76**	70.72	0.71	61.13	0.61	44.16	0.44
PenDigits	84.94	**0.85**	84.57	**0.85**	82.47	0.82	81.18	0.81	60.80	0.61
Soybean	91.11	**0.93**	86.83	0.89	78.82	0.84	83.71	0.83	64.23	0.71
ChessKRvK	36.32	0.33	35.66	0.34	42.79	**0.39**	33.88	0.37	18.59	0.21
LetterRecog	57.37	**0.57**	56.93	**0.57**	41.20	0.41	53.44	0.53	30.41	0.30
Mean	77.43	**0.63**	75.52	0.62	72.54	0.59	72.60	0.61	49.99	0.41

results with respect to ten of the fourteen datasets considered although for one dataset the same result was produced as when CARM classification was used in stand-alone mode and for two datasets the same result was produced as when using Bagging classification. For another three datasets stand-alone CARM produced the best result, although for one dataset the same result was produced as in the case of bagging. For two dataset Bagging classification produced the best result. With respect to the statistical evaluation, no statistically significant difference was detected in the operation of the Binary Tree classification models and the conventional classification models (CARM and Bagging).

The results obtained with respect to the associated run-time experiments are presented in Tables 3.19, 3.20 and 3.21. The tables list both the generation and classification times. From the tables it can be observed, as expected, that the lowest generation and classification times were obtained when using stand-alone classification.

3.5 Summary

A hierarchical ensemble classification model for multi-class classification based on a Binary Tree (BT) structure has been presented in this chapter. Recall that the idea behind the proposed hierarchical classification was to conduct the classification starting in a coarse-grain manner, where examples are allocated to groups of classes, proceeding to a fine-grain manner until class labels can be assigned. To generate such a hierarchical classifier three different distribution techniques were considered: (i) k-means, (ii) *divisive* hierarchical clustering and (iii) data splitting. Three different classification algorithms were used to generate node classifiers: (i) Decision Tree, (ii) Naive Bayes and (iii) CARM. In addition, the use of two different styles of classifier at each node was considered: (i) "stand-alone" and (ii) Bagging ensemble classifier. Two alternative classification strategies: Single Path and Multiple Path were considered. The latter was proposed to address the hierarchical drawback that if an example is mis-classified early on in the process (near the root of the hierarchy) there is no opportunity for recovery. The Multiple Path strategy was realised by utilising Naive Bayes and CARM classifiers, which feature respectively probability and confidence values that can be used to determine where single or multiple paths should be followed (a threshold σ was used to decide whether to follow a single path or not). Three alternative mechanisms to determining the final classification in the case of the Multiple Path strategy were also considered: (i) BIP or BIC class label selection, (ii) NAP or NAC class label selection and (iii) Voting class label selection.

The operation of the proposed Binary Tree hierarchical classification model was compared with two well-established more conventional classification models: (i) stand-alone classification and (ii) Bagging ensemble classification. More specifically, the performance of each type Binary Tree hierarchy considered (Decision Tree, Naive Bayes and CARM)

TABLE 3.18: Average accuracy and AUC results obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM and (iii) Binary Tree hierarchies with CARM classifiers at nodes (K-means&CARM, DS&CARM, and HC&CARM)

Data set	CARM		Bagging		K-means&CARM		DS&CARM		HC&CARM	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Waveform	60.04	0.60	60.76	**0.61**	58.92	0.59	57.02	0.58	61.62	**0.61**
Wine	71.88	0.74	61.48	0.61	78.99	0.81	78.59	0.76	84.28	**0.86**
Nursery	73.94	0.36	73.94	0.36	84.41	**0.43**	86.81	**0.43**	53.07	0.27
Heart	51.70	0.20	45.49	**0.24**	53.76	**0.24**	52.39	0.20	53.35	0.23
PageBlocks	89.99	0.21	89.95	0.21	90.79	0.24	89.77	0.20	73.73	**0.25**
Dermatology	77.00	**0.66**	72.12	0.62	74.92	0.64	60.28	0.46	71.39	**0.66**
Glass	65.05	**0.43**	53.30	0.31	48.46	0.29	61.56	0.31	52.19	0.30
Zoo	94.00	**0.59**	83.00	0.46	88.00	0.52	85.00	0.49	85.00	0.49
Ecoli	49.98	0.12	37.90	0.07	65.15	**0.28**	66.57	0.20	52.17	0.21
Led	67.28	**0.67**	67.09	**0.67**	54.78	0.55	45.53	0.45	35.16	0.35
PenDigits	75.99	0.76	77.09	**0.77**	61.12	0.61	42.68	0.42	50.74	0.51
Soybean	84.01	0.86	73.15	0.77	79.01	0.76	88.97	**0.89**	57.36	0.52
ChessKRvK	17.64	0.07	17.43	0.06	32.99	**0.18**	28.13	0.14	22.50	0.11
LetterRecog	30.91	0.31	30.72	0.31	29.58	0.30	33.12	**0.33**	21.54	0.22
Mean	64.96	**0.47**	60.24	0.43	64.35	0.46	62.60	0.42	55.29	0.40

TABLE 3.19: Run time results (in seconds) obtained using (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) Binary Tree hierarchies with decision trees at nodes (K-means&DT, DS&DT, and HC&DT)

	Generation Time					Classification Time				
Data set	DT	Bagging	K-means&DT	DS&DT	HC&DT	DT	Bagging	K-means&DT	DS&DT	HC&DT
Waveform	0.926	0.901	2.255	1.213	3.816	0.038	0.039	0.390	0.388	0.064
Wine	0.157	0.182	0.181	0.167	0.193	0.001	0.001	0.002	0.001	0.002
Nursery	1.237	1.200	1.609	1.226	5.312	0.151	0.141	0.093	0.098	1.000
Heart	0.189	0.259	0.264	0.204	0.264	0.003	0.006	0.006	0.010	0.003
PageBlocks	0.620	0.736	0.873	0.769	2.870	0.004	0.005	0.009	0.009	0.012
Dermatology	0.210	0.278	0.266	0.228	0.293	0.004	0.011	0.007	0.006	0.004
Glass	0.160	0.217	0.200	0.178	0.228	0.001	0.001	0.003	0.002	0.002
Zoo	0.109	0.128	0.128	0.123	0.134	0.000	0.000	0.001	0.001	0.001
Ecoli	0.179	0.215	0.237	0.201	0.229	0.001	0.001	0.002	0.001	0.002
Led	0.440	0.523	0.568	0.532	0.722	0.007	0.017	0.013	0.012	0.015
PenDigits	1.207	1.308	4.652	2.289	9.253	0.062	0.059	0.091	0.090	0.099
Soybean	0.391	0.457	0.533	0.485	0.614	0.012	0.006	0.006	0.006	0.008
ChessKRvK	2.298	2.123	24.583	2.946	40.116	0.452	0.511	0.365	0.385	0.422
LetterRecog	2.227	1.958	15.037	6.004	34.139	0.185	0.195	0.282	0.242	0.315
Mean	**0.739**	0.749	3.670	1.183	7.013	**0.066**	0.071	0.091	0.089	0.139

TABLE 3.20: Run time results (in seconds) obtained using (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes and (iii) Binary Tree hierarchies with Naive Bayes classifier at nodes (K-means&N, DS&N, and HC&N)

Data set	Generation Time					Classification Time				
	Naive	Bagging	K-means&N	DS&N	HC&N	Naive	Bagging	K-means&N	DS&N	HC&N
Waveform	0.737	0.774	0.839	0.730	1.880	0.002	0.005	0.285	0.226	0.247
Wine	0.202	0.177	0.213	0.167	0.180	0.001	0.001	0.009	0.009	0.009
Nursery	0.974	1.180	1.197	1.048	4.452	0.003	0.011	0.659	0.576	0.609
Heart	0.202	0.216	0.295	0.229	0.226	0.000	0.001	0.045	0.020	0.029
PageBlocks	0.676	0.775	0.763	0.741	2.598	0.001	0.005	0.254	0.266	0.281
Dermatology	0.242	0.296	0.241	0.218	0.248	0.000	0.000	0.028	0.021	0.026
Glass	0.178	0.182	0.209	0.197	0.192	0.000	0.000	0.015	0.011	0.015
Zoo	0.163	0.136	0.146	0.127	0.131	0.000	0.001	0.018	0.017	0.018
Ecoli	0.206	0.208	0.215	0.197	0.211	0.000	0.000	0.031	0.024	0.027
Led	0.529	0.547	0.558	0.530	0.692	0.002	0.004	0.171	0.153	0.189
PenDigits	1.100	1.121	1.359	1.138	5.138	0.006	0.010	0.732	0.507	0.554
Soybean	0.353	0.329	0.458	0.362	0.460	0.001	0.003	0.038	0.047	0.033
ChessKRvK	1.470	1.674	2.021	1.555	17.740	0.006	0.008	1.981	1.254	2.338
LetterRecog	1.398	1.580	1.998	1.481	16.636	0.007	0.011	1.176	0.909	1.126
Mean	**0.602**	0.657	0.751	0.623	3.627	**0.002**	0.004	0.389	0.289	0.393

TABLE 3.21: Run time results (in seconds) obtained using: (i) stand alone CARM classification, (ii) bagging of CARM and (iii) Binary Tree hierarchies with CARM classifiers at nodes (K-means&CARM, DS&CARM, and HC&CARM)

Data set	Generation Time					Classification Time				
	CARM	Bagging	K-means&CARM	DS&CARM	HC&CARM	CARM	Bagging	K-means&CARM	DS&CARM	HC&CARM
Waveform	1.254	1.418	12.296	1.420	12.647	0.002	0.004	0.009	0.009	0.009
Wine	0.389	0.559	0.473	0.460	0.996	0.000	0.000	0.001	0.001	0.001
Nursery	1.142	1.222	1.513	1.186	5.233	0.003	0.004	0.011	0.006	0.011
Heart	0.383	0.624	5.115	0.724	3.982	0.000	0.002	0.002	0.001	0.001
PageBlocks	0.796	0.890	0.978	0.912	2.964	0.001	0.001	0.003	0.002	0.006
Dermatology	0.451	0.660	2.120	0.801	2.918	0.000	0.001	0.001	0.001	0.001
Glass	0.279	0.316	0.406	0.298	0.456	0.001	0.001	0.001	0.001	0.001
Zoo	0.312	0.549	1.270	0.925	1.468	0.000	0.000	0.000	0.001	0.000
Ecoli	0.238	0.276	0.404	0.332	0.393	0.000	0.000	0.001	0.001	0.000
Led	0.581	0.647	0.665	0.593	0.829	0.001	0.015	0.002	0.002	0.008
PenDigits	2.199	2.578	37.004	4.913	57.562	0.008	0.010	0.015	0.013	0.017
Soybean	1.383	2.254	6.182	4.804	11.658	0.005	0.005	0.004	0.002	0.003
ChessKRvK	1.518	1.671	4.068	1.510	19.765	0.004	0.028	0.049	0.026	0.045
LetterRecog	3.510	3.793	203.600	11.160	340.029	0.009	0.017	0.024	0.025	0.061
Mean	**1.031**	1.247	19.721	2.146	32.921	**0.002**	0.006	0.009	0.007	0.012

was compared with the same stand alone classifier and also with Bagging ensemble classification using the same classification algorithms so that a "consistent comparison" was obtained.

From the reported evaluation, and with reference to the evaluation objectives listed at the beginning of Section 3.4. it was demonstrated that:

1. The performance of the binary tree hierarchical ensemble model, was significantly influenced by the adopted data distribution technique. More specifically, it was found that data splitting and k-means were more effective than hierarchical clustering. With respect to efficiency, the most efficient technique for distributing class labels between nodes within the hierarchy, was found to be the data splitting technique.

2. The choice of the base classifiers, to generate the Binary Tree hierarchical ensemble classification model, was also found to significantly affect the classification accuracy of the resulting ensemble model. The most effective and efficient classifier, with which to generate a Binary Tree hierarchical ensemble classification model, was found to be the Naive Bayes classifier.

3. With respect to the comparison between the operation of the two considered classification styles: (i) Stand-alone and (ii) Bagging; using a single (stand-alone) classifier at each tree node outperformed using Bagging. The reason behind the weakness of using Bagging was the insufficient data available for training the classifiers as the process moved down in the tree (recall that a disjoint partition sampling method was used to assign data to the base classifiers to limit the complexity of the proposed model).

4. Following multiple paths within the Binary Tree hierarchy tended to produce a better classification effectiveness, than when following only a single path. More specifically, there was a statistical performance difference between the two strategies with respect to data splitting and hierarchical clustering. While there was no significant difference between the two strategies with respect to k-means. The reason for this is that the successive mis-classification issue occurred more readily with data splitting and hierarchical clustering than in the case of k-means when using the single path classification strategy.

5. With respect to the three considered mechanisms for arriving at a final classification decision in context of the Multiple Path strategies it was demonstrated that the NAP mechanism significantly outperformed the Voting mechanism. No significant difference was detected between the BIP and the NAP mechanisms. Thus, assigning *weight* to the "candidate classes" tended to be more effective than voting as the later mechanism can be significantly affected by votes associated with inaccurate paths.

6. An issue with using confidence values generated by CARM, so as to determine whether one or two branches emanating from a node will be followed, is that it was not always possible to identify both confidence values for a given node's branches. Consequently the unknown confidence values affected the outcome when using a Multiple Path strategy, causing the strategy to be of less value than initially anticipated.

7. A drawback of using clustering techniques to group class labels was identified in that similar classes tended to be grouped together early on in the process so that entire branches ended up dealing with very similar classes. Ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built at each leaf node.

8. The most effective and efficient Binary Tree variation was DS&N, coupled with the Multiple Path strategy. More specifically, using the Naive Bayes algorithm to generate the classifiers at the nodes within the Binary Tree hierarchy, and data splitting to distribute class labels between nodes within the hierarchy.

9. Although the Binary Tree hierarchical classification model improved the classification accuracy with respect to some of the considered data sets, it was demonstrated that there was no statistically significant difference in performance between the proposed Binary Tree hierarchical classification model and the conventional models (stand-alone and ensemble models).

It is interesting to note here that, with respect to the threshold value σ an embedded procedure within a grid-search that selects the best values for the threshold can be adopted, alternatively the value for σ can be user-specified.

The disadvantages of the Binary Tree hierarchical classification model are:

1. The Multiple Path strategy only partially resolves the successive mis-classification problem.

2. The used data distribution techniques was not effective in addressing: (i) the successive mis-classification issue and (ii) generating a high performance hierarchical ensemble model (in comparison with the conventional existing models). Clustering algorithms, even if they work well in grouping class labels, result in grouping similar classes together so that entire branches end up dealing with very similar classes (see point 7 in the above summarisation). While the data splitting technique was found to be a very naive but effective mechanism for distributing class labels between nodes within the hierarchy.

Fundamentally, from the above evaluation, it can be concluded that the Binary Tree structure is not sufficiently expressive to capture the nature of multi-class classification. In order to improve the performance of the Binary Tree hierarchical classification

model, a much more sophisticated structure based on Directed Acyclic Graph (DAG) was considered next to generate the desired hierarchical classification model. The DAG structure sought to address the issues associated with: (i) the grouping of classes at the root and non-leaf nodes and (ii) the resolution of the successive mis-classification problem. This DAG model is therefore considered in the next chapter.

Chapter 4

Rooted Directed Acyclic Graph (*rooted* DAG) for Generating the Hierarchical Classification Model

4.1 Introduction

In this and the following two chapters DAG based hierarchical ensemble classification is considered. The nature of the proposed *rooted* DAG hierarchical classification model is presented in this chapter. As already noted earlier in this thesis, the *rooted* DAG model is founded on the idea of arranging the classifiers into a hierarchical form by utilising a *rooted* DAG structure where each node in the *rooted* DAG holds a classifier. Classifiers at leaves act as binary classifiers while the remaining classifiers (at the root and intermediate nodes) are directed at groupings of class labels. An example *rooted* DAG classifier for four class labels $C = \{a, b, c, d\}$ is presented in Figure 4.1. At the root we classify into four class groups (comprising all possible subset combinations of C where subset size $|C| - 1$). At the next level we classify into three class groups (comprising all possible combinations of *node classes* of size $|nodeclasses| - 1$). Classifiers at the leaves discriminate between two class labels. The main research challenges are then: (i) how best to distribute (organise) the class labels between nodes so as to produce a DAG classifier that generates the most effective classifications and (ii) how to address the successive mis-classification issue imposed by the hierarchical structure. The first issue is discussed further in Section 4.2 where the generation of the *rooted* DAG ensemble model is presented in detail. While the second issue is addressed in Section 4.3 where the operation of the proposed model is presented. Section 4.4 presents an overview of the conducted experiments and the obtained results. Finally, a summary of the chapter is presented in Section 4.5.

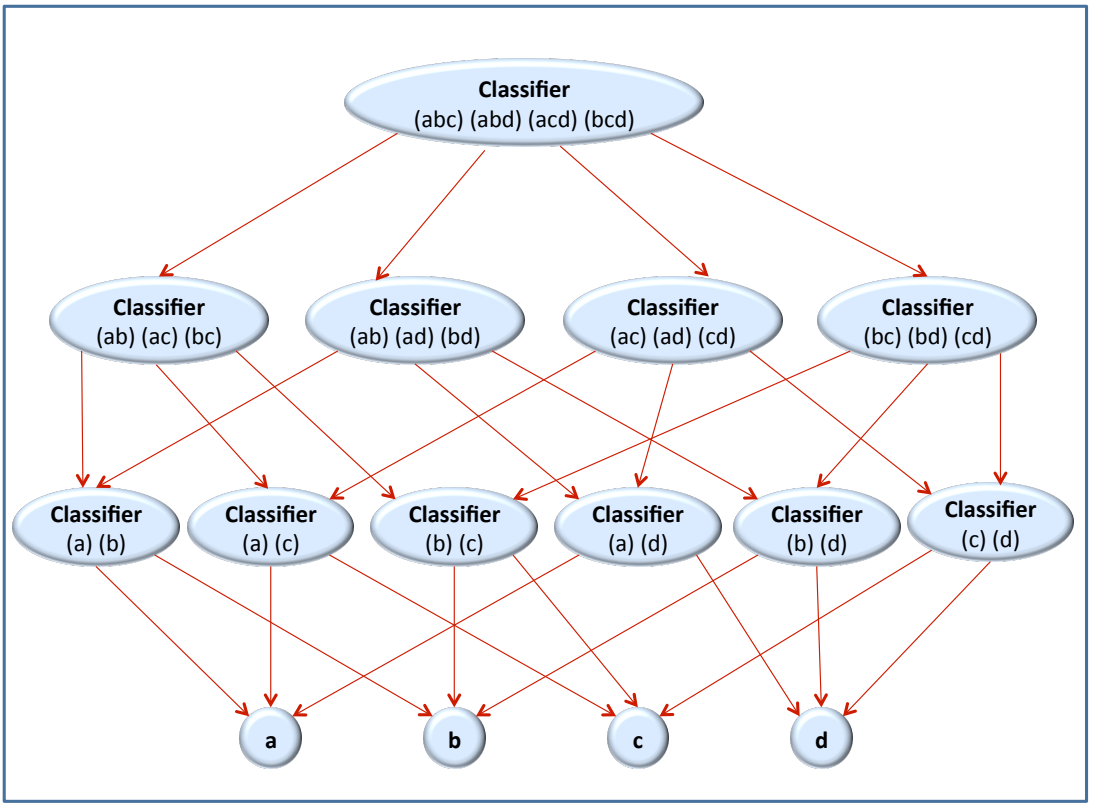


FIGURE 4.1: Rooted DAG example

4.2 Rooted DAG Generation

In this section the generation of the *rooted* DAG hierarchical ensemble classification model is explained. To create the hierarchical ensemble classification model using the *rootedDAG* structure, a classifier needs to be generated for each node in the *rooted* DAG using an appropriate training set. Again, as in the case of the binary tree structure, three different types of classifier were used: (i) Decision tree, (ii) Naive Bayes and (iii) Classification Association Rule Mining (CARM). At start-up the training set D comprises a set of n examples $D = \{r_1, r_2, \ldots . r_n\}$ such that each example has a class label associated with it taken from the set C.

The process requires a class label grouping mechanism. One way of doing this was to apply a clustering mechanism such as k-means (k-means is particularly well suited because the value of k can be pre-specified). But, in the context of binary hierarchies (see Chapter 3), it was found that this clustering approach did not work well because similar classes were grouped together early on in the process so that entire branches ended up dealing with very similar classes, ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built at each leaf node. However, identifying such groups is also not straight-forward. Instead a combination mechanism was used that covers all potential groupings. Starting with the complete class set C at the root of the hierarchy (level $i = 0$), the class groupings

(sub sets) at each level were determined by finding all possible classes combinations of size $|C| - i$ (where i is the level number). As the process proceeded i was increased by one and consequently the "combination size" was decreased by one. The process was continued until the combination size reached two. The process was used to generate Figure 4.1.

It is interesting to note here that a similar approach to rooted DAG class grouping generation, is used with respect to the Apriori frequent item set mining algorithm [1]. Particularly the Apriori-T algorithm [22] in which a particular kind of set enumeration tree called a T-tree is used. The T-tree is essentially a combination of arrays arranged in a tree structure, a data structure sometimes referred to as a Trie. In the top level of the trie we have one item sets, in the second level two item sets and so on. The proposed rooted DAG structure can be argued to have some similarity with this structure.

The number of classifiers needed to be learned in order to generate the *rooted* DAG classification model can be calculated using (4.1) where N is the number of class labels in a given dataset. For example, given a dataset with five class labels 26 classifiers will be required, while if we have four or three classes the number of classifiers to be trained will be 11 and 4 respectively.

$$NumberOfClassifiers = 2^N - N - 1 \tag{4.1}$$

Algorithm 7 presents the generation process in more detail. The input to the algorithm is the training data set D and the set of class labels C. The *rooted* DAG is created in a recursive top down manner starting with $k = |C| - 1$ (where k is the combination size) to $k = 2$ using the function *dagGen*. For each call to *dagGen* the set of size k class combinations, the set C_k, is calculated (line 14). We then loop through this set (line 16) and on each iteration: (i) find the set of training set examples T_i that feature the combination $C_i \in C_k$ (line 17), (ii) generate a classifier G_i using T_i (line 18); (iii) create a new DAG node, *node* (line 19); and (iv) add the new node to the set of accumulated level k nodes so far, $NodeSet$ (line 20). We then loop through the set of current nodes (from the previous iteration) and add a link from each current node $CurrentNode_j$ to the new node *node* whenever the set of class labels associated with the new node (C_i) is included in the set of class labels associated with a current node ($C_i \subset CurrentNodes_j.C$). Finally, if k has not yet reached 2 we repeat (line 27).

4.3 Rooted DAG Operation

Section 4.2 above described the process for generating the hierarchical ensemble classification model using the *rooted* DAG structure. After the model has been generated it is ready for use. In this section the operation of the suggested model is explained. As in the case of the binary tree hierarchies presented in the previous chapter two strategies were considered for classifying individual examples: single path and multiple path. The Single Path strategy is the most straight-forward, and involves following a single path

Algorithm 7 Rooted DAG Generation

```
1: INPUT
2: D = The input training dataset
3: C = The set of Classes featured in D
4: OUTPUT
5: The generated DAG
6:
7: Start
8: k = |C| − 1
9: root = the root node for the DAG
10: dagGen(k, root)
11: End
12:
13: function dagGen(k, CurrentNodes)
14:     C_k = Set of size k combinations in C
15:     NodeSet = {}
16:     for i = 1 to i = |C_k| do
17:         T_i = Set of training examples in D that feature C_i (T_i ⊂ D)
18:         G_i = Classifier for C_i built using training set T_i
19:         node = new Node(G_i, C_i)
20:         NodeSet = NodeSet ∪ new node
21:         for j = 1 to j = |CurrentNodes| do
22:             if C_i ⊂ CurrentNodes_j.C then
23:                 CurrentNodes_j.childNodes = CurrentNodes_j.childNodes ∪ node
24:             end if
25:         end for
26:     end for
27:     if k > 2 then
28:         dagGen(k − 1, NodeSet)
29:     end if
30: end function
```

through the *rooted* DAG, as directed by the individual node classifications, until a leaf node is arrived at. Leaf nodes, as already noted, hold binary classifiers; thus when a leaf node is reached a binary classification can be conducted and a single class label can be assigned to the example. However, as also already noted, the issue with the single path strategy is that if a mis-classification occurs early on in the process there is no opportunity for addressing this situation later on in the process. The Multiple Path strategy was designed to address this problem by allowing more than one path to be followed within the *rooted* DAG. The Multiple Path strategy was realised by utilising Naive Bayes classifiers and Classification Association Rule Miners (CARM), which feature respectively probability and confidence values that can be used to determine where single or multiple paths should be followed. More specifically, more than one path was followed within the *rooted* DAG according to a predefined threshold σ, in the case of Naive Bayes classifiers ($0 \leq \sigma < 1$), while in the case of Classification Association Rule Miners (CARM) ($0 \leq \sigma \leq 100$).

In cases where more than one path was followed we may end up with a number of alternative class labels at the leaf nodes of the DAG, thus we have a set of "candidate class labels". As noted previously, in order to determine a final classification several

mechanisms can be adopted, such as: (i) simply selecting the candidate class associated with the highest "individual" probability (or confidence) value (BIP/BIC) or (ii) generating an accumulated weight for each candidate class and selecting the class associated with the highest accumulated weight (NAP/NAC), or (iii) applying some Voting scheme and selecting the candidate class associated with the highest vote.

The rest of this section is organised as follows: Sub-section 4.3.1 explains the Single Path strategy, while Sub-section 4.3.2 considers the Multiple Path strategy.

4.3.1 Single Path Strategy

As the name suggests, in the Single Path strategy only a "single path" will be followed according to the classification at each DAG node. Algorithm 8 summarises the procedure. The input is a new unseen example, e, to be classified and a pointer ($Root$) to the start of the DAG. The $dagClassify$ procedure is recursive. On each recursion the algorithm is called with two arguments: e, the example to be classified, and a pointer to the current node location in the DAG. The classifier at the current DAG node, $Node.G_i$, is used to classify e (line 12). The resulting classification may be a single class label (in which case we are at a leaf node) or a group of class labels. If the result is a single class then we return this class (line 14). If we have a group of classes $dagClassify$ is called again (line 17) with e and a pointer ($ChildNode$) to the child node associated with the identified class group.

If only a single path is followed within the *rooted* DAG then $N - 1$ classifiers will be evaluated in order to classify a given example, one classifier at each level (where N is the number of class labels in a given dataset).

Algorithm 8 Rooted DAG Single Path Classification

```
 1: INPUT
 2: e = A new unseen example
 3: Root = Start node for the DAG
 4: OUTPUT
 5: The predicted class label c for the input example e
 6:
 7: Start
 8: c = dagClassify(e, Root)
 9: End
10:
11: function dagClassify(e, Node)
12:     C = Classification result for e using classifier Node.G_i
13:     if |C| == 1 then
14:         return c (c ∈ C)
15:     else
16:         ChildNode = child node representing class group C
17:         return (dagclassify(e, ChildNode))
18:     end if
19: end function
```

4.3.2 Multiple Path Strategy

Using the Multiple Path strategy more than one path may be followed as a result of the classification conducted at each current DAG node. With respect to a binary tree structure, where only two branches emanate from each root and body node, using the multiple path strategy all paths greater than a predefined threshold value (σ) were followed. However, when using a DAG structure each node has many branches, if all branches greater than σ are followed this will require the evaluation of a large number of classifiers. More specifically, if all branches feature a probability (or confidence) value greater than σ will be followed, then the number of classifiers to be evaluated in order to classify an example in the worst case (in which all possible paths are explored) is given by:

$$NumberOfClassifiersForAllBranchesStrategy = e\Gamma(N+1,1) - 2N!^{1} \quad (4.2)$$

where N is the number of class labels in a given dataset.

According to Equation 4.2 if N is large number of the classifiers that need to be evaluated will be very large; this means a high classification time. Theoretically it is thus not efficient to follow all possible paths within the *rooted* DAG, and this is evidenced experimentally by the classification times reported later in Section 4.4.

Alternatively, a mechanism for restricting the number of branches to be explored was proposed whereby the maximum number of branches to be explored at each DAG node, whose associated branch value (probability or confidence value as appropriate) was greater than σ, was restricted to some predefined number. Experiments (reported on later in this chapter) were conducted with respect to a maximum of two and three branches, and using all branches. We refer to these three strategies as the *two*, *three* and *all branch* strategies. Intuitively the two branch strategy seemed to be the most appropriate. There were two reasons for this: (i) it allowed for the comparison of the operation of the DAG based hierarchically ensemble classifiers with the operation of the binary tree based hierarchically ensemble classifiers where no more than two paths could be selected at each node, and (ii) it limited the required classification time. Equations 4.3 and 4.4 clarify the second reason. In the worst case the number of classifiers to be evaluated, when using the two branches strategy is given by:

$$NumberOfClassifiersForTwoBranchesStrategy = 2^{(N-1)} - 1 \quad (4.3)$$

where N is the number of class labels in a given dataset.

While the number of classifiers to be evaluated in the worst case, when using the three branches strategy is given by:

[1] Γ (a,x) is the incomplete gamma function, which is a mathematical function used to simplify the following original equation: $NumberOfClassifiersForAllBranchesStrategy = \sum_{i=0}^{N-2} \frac{N!}{(N-i)!}$

$$NumberOfClassifiersForThreeBranchesStrategy = 1/6\left(3^{(N)} - 3\right) \quad (4.4)$$

where N is the number of class labels in a given dataset.

From the above it is clear that following three branches at each DAG node will be computationally more expensive (result in higher classification time) than when following two branches at each DAG node. More specifically, given a dataset with four class labels, the number of classifiers to be trained will be 11 according to Equation 4.1. In this case, following a single path within the *rooted* DAG to classify an example, requires the evaluation of 3 classifiers ($N - 1$); while when following two (or three branches) as a maximum at each DAG node, will require the evaluation of 7 (or 13) classifiers in the worst case. Following all possible paths requires the evaluation of 17 classifiers in the worst case (according to Equation 4.2). The effect on run time is evidenced by the classification times reported later in Section 4.4.

Given the above discussion, following two branches as maximum, at each DAG node, will be discussed and analysed in further detail here. At each DAG node, the two class groups associated with the highest probabilities (highest confidence) are identified, then if their probabilities (confidence values) are greater than σ both branches will be explored, otherwise the branch with the highest associated probability (or confidence) value will be selected.

Algorithm 9 presents the Multiple Path strategy coupled with the normalised accumulated weight mechanism. Note that the algorithm assumes usage of Naive classifiers at each DAG node, the procedure is the same when using CARM but instead of using probability values the associated confidence values are used. The main differences between the Single Path (Algorithm 8) and Multiple Path (Algorithm 9) are: (i) the use of the σ path selection threshold (as part of the input) to decide whether to follow a single branch or two branches among the branches emanating from a node; and (ii) the use of a storage structure $Path$ which, on completion, will contain one or more tuples of the form $\langle c, normProb \rangle$, where c is a class label and $normProb$ is a normalised accumulated Bayesian probability. The algorithm proceeds in a depth first manner and maintains a accumulated Bayesian probability ($accumProb$) and a counter of the number of classifiers that have been invoked ($counter$) as it proceeds (both are set to zero at the start). Once the search is complete the possible classifications arrived at will be contained in $Path$. We select the classification (line 16) with the highest normalised accumulated probability. On each iteration, as the algorithm proceeds, the Bayes classifier at the current node, $Node.G_i$ is applied to r to produce a set of classes C and a set of associated probabilities P. At each node there are four possible outcomes as follows:

1. $|C_1| == 1$ and $p_2 < \sigma$ in which case add $c \in C_1$, and the associated normalised probability value, to $Path$ and return.

2. $|C_1| == 1$ and $p_2 \geq \sigma$ in which case add both $c \in C_1$ and $c \in C_2$, and the associated normalised probability values, to $Path$ and return.

3. $|C_1| \neq 1$ and $p_2 < \sigma$ in which case increment the associated accumulated probability value and the counter for C_1, and continue down the DAG following the path indicated by C_1.

4. $|C_1| \neq 1$ and $p_2 \geq \sigma$ in which case increment the associated accumulated probability values and the counters for both class groups (C_1 and C_2), and continue down the DAG following the paths indicated by C_1 by C_2.

Where: (i) C_1 is the class group in C associated with the highest probability, (ii) p_1 is the Bayesian probability associated with C_1, (iii) C_2 is the class group in C associated with the second highest probability and (iv) $p_2 =$ is the Bayesian probability associated with C_2. Note that, where appropriate, the normalised probability is calculated (lines 27 and 35) by dividing the accumulated probability so far by the number of classifiers that have been invoked.

It is interesting to note that in the case of using confidence values to follow multiple paths within the *rooted* DAG, two or more class groups might have the same confidence value, in this case the solution is simply to choose the class group classified by the rule that appears first in the rule list. Note that with respect to CARM the rules are ordered according to confidence value so that rules with the highest confidence are listed first. If two rules have the same confidence the more general rule, that with the smallest antecedent, will appear first, with more specific rules appearing later. While in the case of using Naive classifiers, the situation where two paths have the same associated probability value is rare, especially when using real datasets, because that means that: (i) the dataset feature the same number of examples for each class (thus a perfectly balanced dataset) and (ii) given a set of attributes in an example to be classified, the probability of each attribute associated with each class is identical. Both situations seem unlikely with respect to real datasets. In the context of the datasets used for evaluation purposes with respect to the work presented in this thesis this situation did not arise. Even if this situation did arise an exception can be set so that all branches that feature the same probability are followed (provided these probabilities are the highest).

The procedure for following a maximum of three branches, at each DAG node, is the same as the procedure included in algorithm 9, except that the three class groups associated with the highest probabilities (highest confidence) are identified instead of only two class groups. The algorithm for following all possible paths within the *rooted* DAG, which feature a value greater than σ, is presented in Appendix E.

Algorithm 10 presents the Multiple Path strategy coupled with the Voting mechanism. The algorithm is very similar to algorithm 9, but much simpler as there is no need to store probability or confidence values during the classification process. Only the individual class labels obtained during traversal of the *rooted* DAG will be added to the $Path$ storage structure (lines 23 and 30). On completion $Path$ will contain a set

Algorithm 9 Rooted DAG Multiple Path Classification Coupled with Normalised Accumulated Probability

INPUT
e = A new unseen example
$Root$ = Start node for the DAG
σ = Path selection threshold
OUTPUT
The predicted class label c for the input example e

GLOBAL VARIABLES
$Path = \{\}$ (Set of identified paths each comprised of: (i) a class label and
(ii) an associated normalised Bayesian probability value)

Start
$accumProb = 0.0$ (Accumulated Bayesian probability start value)
$counter = 0$ (Counter for number of probability values in a followed path)
$dagMultiPathClassify(e, Root, accumProb, counter)$
c = Class label with highest probability value in $Path$
End

function $dagMultiPathClassify(e, Node, accumProb, counter)$
    C = Classification result for e using classifier $Node.G_i$
    P = Bayesian probability values associated with each class group in C
    C_1 = Class group in C associated with highest probability value
    p_1 = Bayesian probability associated with C_1
    C_2 = Class group in C associated with second highest probability value
    p_2 = Bayesian probability associated with C_2
    if $|C_1| == 1$ **then**
        $normProb = (AccumProb + p_1)/(counter + 1)$
        $Path = Path \cup \langle c, normProb \rangle$ $(c \in C_1)$
    else
        $ChildNode$ = child node representing class group C_1
        $dagMultiPathClassify(e, ChildNode, accumProb + p_1, counter + 1)$
    end if
    if $p_2 \geq \sigma$ **then**
        if $|C_2| == 1$ **then**
            $normProb = (AccumProb + p_2)/(countert + 1)$
            $Path = Path \cup \langle c, normProb \rangle$ $(c \in C_2)$
        else
            $ChildNode$ = child node representing class group C_2
            $dagMultiPathClassify(e, ChildNode, accumProb + p_2, counter + 1)$
        end if
    end if
end function

of candidate class labels, the class label with highest occurrences count in $Path$ will be assigned to the example.

With respect to Multiple Path strategy coupled with BIP/BIC the algorithm is very similar to Algorithm 9, however only the probability value produced by the last classifier in a followed path will be added to the $Path$ storage structure. On completion $Path$ will contain a set of candidate class labels, each associated with individual probability value, the class label associated with the highest probability in $Path$ will be assigned to the example.

Algorithm 10 Rooted DAG Multiple Path Classification Coupled with Voting Scheme

```
1: INPUT
2: e = A new unseen example
3: Root = Start node for the DAG
4: σ = Path selection threshold
5: OUTPUT
6: The predicted class label c for the input example e
7:
8: GLOBAL VARIABLES
9: Path = {} Set of candidate class labels resulting from following multiple paths
10: Start
11: dagMultiPathClassify(e, Root)
12: c = Class label with highest occurrences in Path
13: End
14:
15: function dagMultiPathClassify(e, Node, accumProb, counter)
16:     C = Classification result for e using classifier Node.G_i
17:     P = Bayesian probability values associated with each class group in C
18:     C_1 = Class group in C associated with highest probability value
19:     p_1 = Bayesian probability associated with C_1
20:     C_2 = Class group in C associated with second highest probability value
21:     p_2 = Bayesian probability associated with C_2
22:     if |C_1| == 1 then
23:         Path = Path ∪ c
24:     else
25:         ChildNode = child node representing class group C_1
26:         dagMultiPathClassify(e, ChildNode)
27:     end if
28:     if p_2 ≥ σ then
29:         if |C_2| == 1 then
30:             Path = Path ∪ c
31:         else
32:             ChildNode = child node representing class group C_2
33:             dagMultiPathClassify(e, ChildNode)
34:         end if
35:     end if
36: end function
```

4.4 Experiments and results

In this section we present an overview of the adopted experimental set up and the evaluation results obtained. The effectiveness of the suggested *rooted* DAG classification model was evaluated using twelve different data sets (with various numbers of class labels) taken from the UCI machine learning repository. For comparison purposes alternative forms of classification were also applied to the data sets as follows:

1. A number of "stand alone" classifiers, namely: **Naive Bayes**, **Decision tree**, and **CARM**. The objective being to compare the operation of the proposed *rooted* DAG model with the operation of single conventional models. Other forms of single classification model could have been selected but Naive Bayes, Decision tree and CARM were chosen because these were also used in the context of the evaluation of

the binary tree based hierarchical ensemble classification as described in Chapter 3.

2. A **Bagging** ensemble using a combination of three classifiers, again Naive Bayes, Decision tree and CARM were used as the base classifiers. The objective being to compare the operation of the proposed *rooted* DAG model with alternative forms of ensembles.

3. A **One-Versus-One** (OVO) classification mechanism using support vector machines as the base classifiers. The objective being to compare the proposed *rooted* DAG model with a classification mechanism founded on the use of a set of binary classifiers for solving the multi-class classification problems. The well known LibSVM [18] implementation was used, with the Gaussian Radial Basis Function (RBF) kernel. In order to obtain the best performance the optimal values for the parameters (C and γ) were selected based on a grid search with cross validation, as recommended in [18]. Table 4.1 presents the optimal values of C and γ for each of the considered evaluation datasets.

TABLE 4.1: The optimal values for C and γ with respect to the fourteen considered evaluation datasets

DataSet	Classes	C	γ
Waveform	3	8	0.0039
Wine	3	2	0.0039
Nursery	5	2	1.0000
Heart	5	1	0.5000
PageBlocks	5	16	1.0000
Dermatology	6	256	0.0078
Glass	7	8	0.2500
Zoo	7	32	0.2500
Ecoli	8	1	0.5000
Led	10	1	0.0625
PenDigits	10	1	0.5000
Soybean	15	4	0.0310
ChessKRvK	18	8	8.0000
LetterRecog	26	1	2.0000

The results obtained are presented in the following sections as follows: Section 4.4.1 presents the results obtained using the Single Path strategy with respect to the three alternative classification algorithms. Section 4.4.2 presents the results obtained using the *rooted* DAG ensemble classification model coupled with the Multiple Path strategy. Section 4.4.3 provides a comparison between the *rooted* DAG Single Path and Multiple Path strategies. Section 4.4.4 considers the results of using conventional methods, standalone, Bagging, and OVO classification, compared with the results obtained from the *rooted* DAG model.

4.4.1 Single Path Strategy Experiments and Results

This section presents the results obtained using the Single Path strategy coupled with the three alternative classification algorithms for generating the node classifiers in the *rooted* DAG: Decision Tree, Naive Bayes and CARM. Table 4.2 presents the results in terms of average accuracy, and average AUC. Note here that the results presented in the table references only eleven data sets, the reason for this is that when using either Decision Tree or CARM classifiers as the base classifiers the computational resource required is greater than when using Naive Bayes classifiers[2]. When Naive Bayes classifiers are used the *rooted* DAG model can be applied to datasets featuring up to fifteen class labels because Bayes classification requires less computational resource than when Decision Tree or CARM classification is used. From the table it can be observed that best results are obtained for the majority of the datasets considered when using Naive Bayes classifiers for generating the DAG node classifiers. More specifically, the Naive Bayes DAG produced the best results with respect to nine of the eleven data sets considered (Wave Form, Wine, Nursery, Heart, Dermatology, Glass, Zoo, Ecoli, and Pen Digits). While the Decision tree DAG produced the best results for two of the datasets considered (Page Blocks, and Led). According to the conducted statistical tests there was a statistically significant difference in operation between the three considered classification models; Naive Bayes DAG performed statistically better than both Decision tree DAG and CARM DAG, while no statistically significant difference was observed between Decision tree DAG and CARM DAG.

Table 4.2: Average Accuracy and AUC values obtained using Decision Tree, Naive Bayes and CARM to generate a rooted DAG classification model

Data set	Classes	Decision Tree (DAG)		Naive Bayes DAG		CARM DAG	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	48.42	0.49	**77.00**	**0.77**	68.54	0.69
Wine	3	56.78	0.51	**95.08**	**0.95**	86.26	0.86
Nursery	5	33.35	0.20	**90.26**	**0.45**	86.81	0.43
Heart	5	59.29	0.34	**55.91**	**0.35**	53.76	0.20
PageBlocks	5	**93.20**	**0.54**	92.69	0.52	89.77	0.20
Dermatology	6	44.78	0.39	**87.23**	**0.85**	79.62	0.71
Glass	7	48.06	0.24	**69.81**	**0.46**	61.71	0.36
Zoo	7	74.09	0.42	**92.18**	**0.58**	88.00	0.52
Ecoli	8	78.93	0.33	**84.43**	**0.41**	32.12	0.24
Led	10	**85.47**	**0.85**	75.66	0.76	40.06	0.40
PenDigits	10	37.93	0.38	**83.58**	**0.83**	46.76	0.47
Mean		60.03	0.43	**82.17**	**0.63**	66.67	0.46

[2]All experiments were conducted using a 2.7 GHz Intel Core i5 with 16 GB 1333 MHz DDR3 memory, running OS X 10.9.2 (13C64).

With respect to using CARM as a base classifiers to generate the desired model, it was necessary to use a very low value for the confidence threshold (τ) so as to generate the required internal classifiers ($\tau = 16$). The confidence threshold value of $\tau = 40$, that was used previously with respect to the binary tree hierarchies discussed earlier in chapter 3 to generate the internal classifiers, was found not to be the best threshold with respect to the *rooted* DAG model. More specifically, using $\tau = 40$ results in no classification rules being generated for many internal classifiers within the *rooted* DAG, and consequently low classification accuracies were recorded. Therefore the confidence threshold value was reduced to $\tau = 16$ (see Appendix F).

The reasons behind the low confidence values generated by the internal classifiers are:

1. That the internal classifiers within the *rooted* DAG distinguish between large numbers of class combinations, compared to the binary tree structure where the internal classifiers distinguish between only two groups of class labels.

2. The support (frequency) of a rule is given by the number of examples in the training data for which the rule is found to apply [24] and the confidence of the rule is the ratio of its support to the support for its antecedent [24]. The combination procedure results in a high support value for the antecedents of rules, consequently the resulting confidence values tend to be low.

The results obtained for the run-time experiments with respect to the three node classifier generators considered with respect to the *rooted* DAG model are presented in Table 4.3. The table presents the generation and classification times for each. From the table it can be observed that the lowest generation and classification times were obtained when using Naive Bayes to generate the *rooted* DAG ensemble classification model.

From the above discussion, we can conclude that the choice of the base classifiers, to generate the *rooted* DAG ensemble classification model, can significantly affect the classification accuracy of the resulting ensemble model. The most effective and efficient classifier, to generate the *rooted* DAG ensemble classification model, was found to be a Naive Bayes classifier.

4.4.2 Multiple Path Strategy Experiments and Results

This section presents the results obtained using the *rooted* DAG ensemble classification model coupled with the proposed Multiple Path strategy. As noted earlier, the Multiple Path strategy was realised using Naive Bayes and CARM classifier generators for the node classifiers, because these featured probability or confidence values that could be used to determine whether single or multiple paths should be followed. Consequently, this section is divided into three parts as follows: (i) Part 1 which presents the conducted experiments and the obtained results when following multiple paths using Naive Bayes classifiers, (ii) Part 2 which presents the conducted experiments and the obtained results

TABLE 4.3: Run time results (in seconds) obtained using Decision Tree, Naive Bayes, and CARM to generate a rooted DAG classification model

Data set	Generation Time			Classification Time		
	Decision Tree	Naive Bayes	CARM	Decision Tree	Naive Bayes	CARM
WaveForm	2.726	0.199	2.494	0.071	0.000	0.005
Wine	0.238	0.189	0.682	0.002	0.000	0.000
Nursery	15.222	5.982	5.547	0.266	0.012	0.009
Heart	0.525	0.333	3.788	0.006	0.001	0.000
PageBlocks	3.743	2.510	3.404	0.017	0.008	0.003
Dermatology	0.868	0.445	8.542	0.008	0.001	0.001
Glass	1.006	0.539	2.281	0.005	0.001	0.000
Zoo	0.691	0.491	14.938	0.002	0.001	0.000
Ecoli	2.723	1.032	0.992	0.005	0.001	0.001
Led	505.184	33.701	25.451	0.030	0.011	0.022
PenDigits	4256.904	264.369	2402.572	0.569	0.039	0.055
Mean	435.439	**28.163**	224.608	0.089	**0.007**	0.009

when following multiple paths using CARM as the base classifier and (iii) Part 3 which presents a comparison between the two. The objectives of the experiments were: (i) to observe the effectiveness of following multiple paths within the *rooted* DAG, (ii) to determine the effect that the number of branches explored had on the classification time (thus efficiency), (iii) to identify the most effective mechanism for selecting the final class label for a given, previously unseen, example and (iv) to determine the most effective classifier to be utilised with respect to the proposed Multiple Path strategy.

Part1: Using Naive Bayesian Probability Values for Following Multiple Paths Within the Rooted DAG. The conducted experiments and the obtained results when following multiple paths and utilising Naive Bayes probability values are presented here. As noted earlier, for reasons of both efficiency and effectiveness it is preferable to follow a maximum of up to two branches at each DAG node (the two branch strategy). In this section the obtained results when following a maximum of two, three and all branches (as indicated by the value for σ) are presented. The objectives of the experiments were: (i) to observe the effectiveness of following multiple paths within the *rooted* DAG and (ii) to determine the effect that the number of branches explored had on the classification time (thus efficiency). This section also presents a comparison between the three different proposed mechanisms for arriving at a final classification result: (i) Normalised Accumulated Probability NAP, (ii) Best Individual Probability BIP and (iii) Voting. The objective here is to identify the most effective mechanism for selecting the final class label for a given, previously unseen, example.

Following two branches, as maximum, at each DAG node. The results obtained when following up to two branches, at each DAG node, where the branch values are greater than σ are presented here. Experiments were conducted using NAP, Voting

and BIP to identify the most appropriate value of σ in each case. Some detail concerning these experiments are presented in Appendix F. Using the NAP class label selection mechanism it was found that $\sigma = 0.7 \times 10^{-4}$ produced the best performance. Using the Voting class label mechanism it was found that $\sigma = 0.1 \times 10^{-5}$ produced the best performance and using the BIP class label mechanism it was found that $\sigma = 0.5 \times 10^{-4}$ produced the best performance.

A comparison between the NAP, BIP and Voting mechanisms is presented in Table 4.4 (best results highlighted in bold font). From the table it can be observed that the results of the three mechanisms are very similar. More specifically, for seven datasets (WaveForm, Wine, Nursery, Dermatology, PageBlocks, PenDigits, and Soybean) the same AUC value was obtained regardless of which class selection mechanism was adopted. For three datasets (Heart, Ecoli, and Led) the Voting mechanism produced the best AUC value, although for one data set (Led) the same result was produced by NAP. For one dataset (Glass) the NAP mechanism produce the best AUC results. For another one dataset (Glass) the BIP mechanism produce the best AUC results. The conducted statistical evaluation demonstrated that no significant difference in performance between the three mechanisms.

TABLE 4.4: Comparison between: (i) NAP, (ii) BIP and (iii) Voting mechanisms for determining the final resulting class label with respect to rooted DAG

Data set	Classes	NAP ($\sigma = 0.7 \times 10^{-4}$)		Voting ($\sigma = 0.1 \times 10^{-5}$)		BIP ($\sigma = 0.5 \times 10^{-4}$)	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	**77.00**	**0.77**	**77.00**	**0.77**	**77.00**	**0.77**
Wine	3	**95.08**	**0.95**	**95.08**	**0.95**	**95.08**	**0.95**
Nursery	5	**90.28**	**0.45**	90.26	**0.45**	**90.28**	**0.45**
Heart	5	55.37	0.35	**57.98**	**0.36**	56.19	0.35
PageBlocks	5	**92.65**	**0.52**	**92.65**	**0.52**	**92.65**	**0.52**
Dermatology	6	**87.23**	**0.85**	87.18	**0.85**	86.94	**0.85**
Glass	7	**72.99**	**0.51**	70.29	0.45	72.11	0.50
Zoo	7	92.18	0.58	91.18	0.57	**93.18**	**0.59**
Ecoli	8	82.56	0.38	**84.38**	**0.39**	82.56	0.38
Led	10	**75.56**	**0.76**	75.47	**0.76**	75.41	0.75
PenDigits	10	**83.58**	**0.83**	**83.58**	**0.83**	**83.58**	**0.83**
Soybean	15	**90.75**	**0.92**	**90.75**	**0.92**	**90.75**	**0.92**
Mean		82.936	**0.656**	**82.983**	0.652	82.978	0.655

Three Branch Strategy. The results obtained when following up to three branches at each DAG node (where each branch has a value greater than σ) are presented here. Again, as in the case of following a maximum two branches at each DAG node, a range of alternative values for σ were considered. As a result it was found that $\sigma = 0.1 \times 10^{-4}$ produced the best average AUC value for the considered evaluation datasets. For completeness these results are included in Appendix F.

All Branch Strategy. The results obtained when following all possible branches with a value greater than σ at each DAG node are presented here. Again, as in the case of following up to two and three branches at each DAG node, a range of alternative threshold values for σ were considered (see Appendix F) and it was found that $\sigma = 0.1 \times 10^{-4}$ produced the best average AUC value for the considered evaluation datasets.

Effectiveness and efficiency comparison between the maximum number of branches to be followed at each DAG node. A comparison between the two, three and all branch strategies is presented here. The best results, are presented in Table 4.5 (best AUC results highlighted in bold font). From the table, and according to the provided average AUC values, it can be observed that the three and all branches strategies produced better AUC values than when up to two branches were followed. In addition it can be observed that there is no significant difference between the AUC values obtained when using either the three or all branches strategies. Although we can notice an improvement in terms of the obtained AUC values when comparing the two and three (or all branches) strategies. The efficiency associated with the three or all branch strategies is however an important consideration. Table 4.6 presents the run times with respect to the two, three and all branch strategies (greater than a predefined threshold value) within the *rooted* DAG. From the table it can be clearly observed, as expected, that the classification time increases dramatically when the number of branches to be explored increases.

Table 4.5: Average accuracy and average AUC results obtained using the two, three and all branches strategies when generating a rooted DAG (Naive Bayes)

		Two branches		Three branches		All branches	
Data set	Classes	Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	77.00	**0.77**	77.00	**0.77**	77.00	**0.77**
Wine	3	95.08	**0.95**	95.08	**0.95**	95.08	**0.95**
Nursery	5	90.28	0.45	88.94	**0.58**	89.48	**0.58**
Heart	5	55.37	0.35	54.73	**0.37**	54.94	**0.37**
PageBlocks	5	92.65	0.52	91.69	**0.53**	91.85	**0.53**
Dermatology	6	87.23	**0.85**	86.37	0.84	87.23	**0.85**
Glass	7	72.99	**0.51**	59.09	**0.51**	61.87	**0.51**
Zoo	7	92.18	0.58	93.18	**0.59**	93.18	**0.59**
Ecoli	8	82.56	**0.38**	68.55	0.33	72.45	0.34
Led	10	75.56	**0.76**	75.41	**0.76**	75.41	**0.76**
PenDigits	10	83.58	**0.83**	83.58	**0.83**	83.58	**0.83**
Soybean	15	90.75	**0.92**	90.75	**0.92**	90.75	**0.92**
Mean		82.94	0.66	80.36	**0.67**	81.07	**0.67**

As noted earlier in Section 4.3.2 it is preferable to restrict the number of branches to be followed to a maximum of two so as to maintain the required run time within reasonable limits. To be more confident about this decision, a Friedman test was applied to examine if the AUC results produced when adopting either the three or all branch

TABLE 4.6: Run time results (in seconds) obtained when using the two, three and all branch strategies when generating a rooted DAG (Naive Bayes)

Data set	Classes	Two branches	Three branches	All branches
WaveForm	3	0.008	0.009	0.009
Wine	3	0.007	0.009	0.009
Nursery	5	0.595	0.621	0.622
Heart	5	0.015	0.020	0.021
PageBlocks	5	0.266	0.278	0.299
Dermatology	6	0.020	0.028	0.050
Glass	7	0.016	0.026	0.060
Zoo	7	0.009	0.019	0.039
Ecoli	8	0.031	0.090	1.138
Led	10	0.261	1.318	1407.375
PenDigits	10	0.723	0.751	2235.709
Soybean	15	0.068	0.082	259.477
Mean		0.168	0.271	325.401

strategies were statistically different than those obtained when using the two branch strategy. According to the conducted Friedman test there was no statistically significant difference between the different strategies. Based on the above reported effectiveness and efficiency results, it was concluded that the two branch strategy was the most appropriate. This was the strategy adopted with respect to the remaining experiments reported on in this chapter.

Part 2: Using CARM Confidence Values for Following Multiple Paths Within the Rooted DAG The results of following multiple paths within the *rooted* DAG structure by utilising the confidence values generated when using CARM are presented here. Because the foregoing section established that it is not significantly effective nor efficient to follow more than two branches at each DAG node, the results presented in this section have all been generated using the two branch strategy. A range of alternative values for σ were considered (see Appendix F) and it was found that $\sigma = 50$ produced the best performance for the considered evaluation datasets.

Part 3: Comparison Between Using Probability Values and Confidence Values for Following Multiple Paths Within the Rooted DAG. A comparison between using Naive Bayesian probability values and CARM confidence values for following multiple paths within the *rooted* DAG is presented here. The objective of the comparison is to determine the most effective classifier to be utilised with respect to the proposed Multiple Path strategy. Table 4.7 present the results obtained. From the table it can be clearly observed that utilising Naive Bayesian probability values significantly outperforms utilising CARM confidence values in the context of the two path strategy. Again, as the case of single path strategy, Naive Bayes classifiers are the best choice for generating the *rooted* DAG, compared to CARM and decision tree classifiers.

From the above it was concluded that: (i) no significant difference in performance between the three mechanisms for arriving at a final classification decision (Voting, BIP and NAP) when following multiple paths within the DAG and (ii) the most effective classifier to be utilised with respect to the Multiple Path strategy was found to be Naive Bayes classifier.

TABLE 4.7: Average Accuracy and AUC values obtained using Naive Bayes and CARM to generate rooted DAG classification models coupled with the two branch strategy

Data set	Classes	Naive Bayes DAG		CARM DAG	
		Acc.	AUC	Acc.	AUC
Waveform	3	**77.00**	**0.77**	68.54	0.69
Wine	3	**95.08**	**0.95**	86.26	0.86
Nursery	5	**90.28**	**0.45**	86.81	0.43
Heart	5	**55.37**	**0.35**	53.76	0.20
PageBlocks	5	**92.65**	**0.52**	89.77	0.20
Dermatology	6	**87.23**	**0.85**	79.62	0.71
Glass	7	**72.99**	**0.51**	61.71	0.36
Zoo	7	**92.18**	**0.58**	88.00	0.52
Ecoli	8	**82.56**	**0.38**	32.42	0.25
Led	10	**75.56**	**0.76**	40.06	0.43
PenDigits	10	**83.58**	**0.83**	46.76	0.47
Mean		**82.23**	**0.63**	66.70	0.47

4.4.3 Comparison Between Single and Multiple Path Strategies

The objective of the comparison between the Single Path and Multiple Path strategies was to determine whether following more than one path within the *rooted* DAG classification model could address the successive mis-classification issue noted earlier.

Commencing with a comparison of the Single and Multiple Path strategies with respect to the Naive Bayes classification. From experiments conducted previously, and presented above, the two path strategy was adopted for this purpose. Although, again from the above presented experiments, as a general rule $\sigma = 0.7 \times 10^{-4}$ had been found to produce the best performance for most of the datasets considered, a specific best value for σ can be identified for each data set. Table 4.8 presents the average accuracy and AUC results obtained using the two branch strategy, in comparison with using a single path (best AUC values highlighted in bold). From the table it can be observed that by using the Multiple Path strategy the operation of the proposed *rooted* DAG classification model is such that the classification accuracy with respect to one of the twelve data sets considered (Glass) is improved. For one dataset (Ecoli) using the Single Path strategy produced the best result. For the remaining ten data sets the same AUC results were obtained regardless of whether a Single Path or Multiple Path strategy was adopted.

TABLE 4.8: Average Accuracy and AUC results obtained using Naive Bayes coupled with either a Single or a Multiple Path strategy

Data set	Classes	Naive Bayes			
		Single Path		Multiple Path ($\sigma = 0.7 \times 10^{-4}$)	
		Acc.	AUC	Acc.	AUC
Waveform	3	77.00	**0.77**	77.00	**0.77**
Wine	3	95.08	**0.95**	95.08	**0.95**
Nursery	5	90.26	**0.45**	90.28	**0.45**
Heart	5	55.91	**0.35**	55.57	**0.35**
PageBlocks	5	92.69	**0.52**	92.69	**0.52**
Dermatology	6	87.23	**0.85**	87.23	**0.85**
Glass	7	69.81	0.46	72.99	**0.51**
Zoo	7	92.18	**0.58**	92.18	**0.58**
Ecoli	8	84.43	**0.41**	82.56	0.38
Led	10	75.66	**0.76**	75.56	**0.76**
PenDigits	10	83.58	**0.83**	83.58	**0.83**
Soybean	15	90.75	**0.92**	90.75	**0.92**
Mean		82.88	0.65	82.94	**0.66**

Regarding the use of CARM classifiers to generate the proposed *rooted* DAG model, Table 4.9 presents the average accuracy and AUC results obtained when adopting either a single or a multiple path strategy within a *rooted* DAG (best AUC values highlighted in bold). From the table it can be observed that by using the multiple path strategy the operation of the proposed *rooted* DAG classification model is such that the classification accuracy with respect to two of the eleven data sets considered (Ecoli, and Led) is improved. For the remaining nine datasets the same AUC results were obtained regardless of which strategy was adopted.

With respect to the conducted statistical evaluation, it was found that following multiple paths within the rooted DAG classification model did not produce a statistically significant difference in classification effectiveness than when following only a single path. The reason for this is that the combination technique used to distribute classes between nodes in the DAG resulted in well-defined class labels at each DAG node, unlike in the case where clustering algorithms were used with respect to the Binary Tree hierarchical classification model; consequently the number of mis-classifications is less and the effect of following multiple paths within the DAG is not highly significant.

4.4.4 Comparison Between the Rooted DAG Ensemble Classification Model and Conventional models

In this section a comparison between the proposed *rooted* DAG classification model and conventional classification models is presented. In order to conduct a consistent comparison between the *rooted* DAG and existing conventional models a comparison

TABLE 4.9: Average Accuracy and AUC results obtained using CARM coupled with either a single or a multiple path strategy

Data set	Classes	CARM			
		Single Path		Multiple Path ($\sigma = 50$)	
		Acc.	AUC	Acc.	AUC
Waveform	3	68.54	**0.69**	68.54	**0.69**
Wine	3	86.26	**0.86**	86.26	**0.86**
Nursery	5	86.81	**0.43**	86.81	**0.43**
Heart	5	53.76	**0.20**	53.76	**0.20**
PageBlocks	5	89.77	**0.20**	89.77	**0.20**
Dermatology	6	79.62	**0.71**	79.62	**0.71**
Glass	7	61.71	**0.36**	61.71	**0.36**
Zoo	7	88.00	**0.52**	88.00	**0.52**
Ecoli	8	32.12	0.24	39.51	**0.25**
Led	10	40.06	0.40	40.06	**0.43**
PenDigits	10	46.76	**0.47**	46.76	**0.47**
Mean		66.67	0.46	66.70	**0.47**

was conducted using the same classifier generator in each case, Three set of experiments are reported on here: (i) comparison between the operation of a stand alone decision tree classifier, bagging of decision trees and a decision trees DAG model; (ii) comparison between the operation of a stand alone CARM classifier, bagging of CARM and CARM DAG model; and (iii) comparison between the operation of a stand alone Naive Bayes classification, Bagging of Naive Bayes classifiers and Naive Bayes DAG classification. In addition, a "non-consistent" comparison between Naive Bayes DAG, because the foregoing sections has already established that the Naive Bayes DAG produced the best performance, and OVO SVM was conducted. The objective of this last comparison was to compare the suggested model with one of the state of the art methods for multi-class classification. For the comparison the two branch strategy was used through out (see previous discussion).

Starting with the comparison between "stand-alone" decision tree classification, bagging of decision trees, and the proposed *rooted* DAG with decision tree classifiers at each node. Table 4.10 presents the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). From the table it can be observed that the proposed *rooted* DAG classification model produces an improved classification accuracy with respect to four of the eleven datasets considered (Nursery, Heart, PageBlocks, and Led). In the remaining seven cases, the stand-alone decision tree classifier produced the best result, although for one dataset (Zoo) the same result was produced as in the case of the bagging ensemble classification. With respect to the statistical evaluation, no statistically significant difference was detected in the operation of the Decision Tree DAG and the Decision Tree classification (both stand-alone and Bagging).

The results obtained for a series of run-time experiments that were also conducted are presented in Table 4.11. The table includes both the generation and classification times. From the table it can be observed, as expected, that the lowest generation and classification times were obtained when using stand-alone decision tree classification.

TABLE 4.10: Average accuracy and AUC results obtained when using: (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) DAG classification with decision trees at nodes

Data set	Classes	Decision Tree		Bagging		DAG	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	**53.72**	**0.54**	53.36	0.53	48.42	0.49
Wine	3	**73.86**	**0.73**	69.91	0.69	56.78	0.51
Nursery	5	5.15	0.03	32.71	0.16	**33.35**	**0.20**
Heart	5	48.80	0.28	51.15	0.28	**59.29**	**0.34**
PageBlocks	5	92.55	0.49	92.23	0.47	**93.2**	**0.54**
Dermatology	6	**57.53**	**0.57**	43.95	0.39	44.78	0.39
Glass	7	**64.50**	**0.40**	62.27	0.36	48.06	0.24
Zoo	7	**89.00**	**0.53**	87.27	**0.53**	74.09	0.42
Ecoli	8	78.07	**0.34**	73.61	0.31	**78.93**	0.33
Led	10	74.72	0.74	74.06	0.74	**85.47**	**0.85**
PenDigits	10	**76.84**	**0.77**	72.64	0.72	37.93	0.38
Mean		**64.98**	**0.49**	64.83	0.47	60.03	0.43

TABLE 4.11: Run time results (in seconds) obtained using (i) stand alone decision tree classification, (ii) bagging of decision trees and (iii) DAG classification with decision trees at nodes

Data set	Generation Time			Classification Time		
	Decision Tree	Bagging of Trees	DAG	Decision Tree	Bagging of Trees	DAG
WaveForm	0.926	0.901	2.726	0.038	0.039	0.071
Wine	0.157	0.182	0.238	0.001	0.001	0.002
Nursery	1.237	1.200	15.222	0.151	0.141	0.266
Heart	0.189	0.259	0.525	0.003	0.006	0.006
PageBlocks	0.620	0.736	3.743	0.004	0.005	0.017
Dermatology	0.210	0.278	0.868	0.004	0.011	0.008
Glass	0.160	0.217	1.006	0.001	0.001	0.005
Zoo	0.109	0.128	0.691	0.000	0.000	0.002
Ecoli	0.179	0.215	2.723	0.001	0.001	0.005
Led	0.440	0.523	505.184	0.007	0.017	0.030
PenDigits	1.207	1.308	4256.904	0.062	0.059	0.569
Mean	**0.494**	0.541	435.439	**0.025**	0.026	0.089

With respect to the comparison between "stand-alone" Naive Bayes classification, bagging of Naive Bayes classifiers and the Naive Bayes DAG models Table 4.12 presents

the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). Recall that the results presented with respect to the DAG are the multiple path results when the two branch strategy is adopted. From the table it can be observed that the proposed *rooted* DAG classification model improves classification accuracy with respect to seven of the twelve datasets considered (WaveForm, Heart, PageBlocks, Dermatology, Glass, Ecoli, and Led), although for three datasets (WaveForm, PageBlocks, and Led) the same result was produced as when Naive Bayes classification was used in stand-alone mode and with respect to bagging. For one dataset (Dermatology) the same result as that obtained with respect to stand alone Naive Bayes classification was obtained. For another three datasets (Wine, PenDigits, Soybean) the stand-alone Naive Bayes classifier produced the best result although for one dataset (PenDigits) the same result was produced when bagging was used. For two datasets (Nursery, and Zoo) bagging produced the best results. With respect to the statistical evaluation, no statistically significant difference was detected in the operation of the proposed Naive Bayes DAG and the Naive Bayes classification.

The results obtained with respect to the associated run-time experiments are presented in Table 4.13. The table lists both the generation and classification times. From the table it can be observed, again as expected, that the lowest generation and classification times were obtained when using stand-alone classification.

TABLE 4.12: Average accuracy and AUC obtained when using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes classifiers and (iii) DAG classification with Naive Bayes classifiers at nodes

Data set	Classes	Naive Bayes		Bagging of Naive Bayes		Naive bayes DAG	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	77.04	**0.77**	**77.06**	**0.77**	77.00	**0.77**
Wine	3	**95.67**	**0.96**	93.72	0.94	95.08	0.95
Nursery	5	90.22	0.45	89.96	**0.46**	**90.28**	0.45
Heart	5	54.60	0.34	51.28	0.30	**55.57**	**0.35**
PageBlocks	5	**92.69**	**0.52**	92.62	**0.52**	**92.69**	**0.52**
Dermatology	6	86.66	**0.85**	81.00	0.81	**87.23**	**0.85**
Glass	7	67.83	0.49	55.28	0.46	**72.99**	**0.51**
Zoo	7	92.27	0.59	**94.27**	**0.62**	93.18	0.59
Ecoli	8	81.70	0.38	82.56	0.39	**84.43**	**0.41**
Led	10	**75.59**	**0.76**	75.50	**0.76**	75.56	**0.76**
PenDigits	10	**84.94**	**0.85**	84.57	**0.85**	83.58	0.83
Soybean	15	**91.11**	**0.93**	86.83	0.89	90.75	0.92
Mean		82.53	**0.66**	80.39	0.65	**83.20**	**0.66**

Because of the effectiveness and efficiency of the Naive Bayes DAG model, compared to decision trees and CARM DAGs, a comparison between the operation of the Naive Bayes DAG and OVO SVM, was also conducted. Table 4.14 presents the results obtained

TABLE 4.13: Run time results (in seconds) obtained using: (i) stand alone Naive Bayes classification, (ii) bagging of Naive Bayes classifiers and (iii) DAG classification with Naive Bayes classifiers at nodes

Data set	Generation Time			Classification Time		
	Naive Bayes	Bagging	DAG	Naive Bayes	Bagging	DAG
Waveform	0.737	0.774	0.199	0.002	0.005	0.009
Wine	0.202	0.177	0.189	0.001	0.001	0.009
Nursery	0.974	1.180	5.982	0.003	0.011	0.595
Heart	0.202	0.216	0.333	0.000	0.001	0.015
PageBlocks	0.676	0.775	2.510	0.001	0.005	0.266
Dermatology	0.242	0.296	0.445	0.000	0.000	0.020
Glass	0.178	0.182	0.539	0.000	0.000	0.016
Zoo	0.163	0.136	0.491	0.000	0.001	0.009
Ecoli	0.206	0.208	1.032	0.000	0.000	0.031
Led	0.529	0.547	33.701	0.002	0.004	0.261
PenDigits	1.100	1.121	264.369	0.006	0.010	0.723
Soybean	0.353	0.329	1520.706	0.001	0.003	0.068
Mean	**0.464**	0.495	152.541	**0.001**	0.003	0.169

in terms of average accuracy and average AUC (best results highlighted in bold font). Again, recall that the results presented with respect to the DAG are the multiple path results obtained when using the two branches strategy. From the table it can be observed that the Naive Bayes DAG produced the best classification accuracy with respect to eight of the twelve datasets considered (Wine, Heart, PageBlocks, Dermatology, Glass, Zoo, Ecoli, Led, Soybean), although for one dataset (Led) the same result was produced using OVO SVM. In the remaining four cases, the OVO SVM produced the best result. According to the statistical test result, no statistically significant difference was detected in the operation of the Naive Bayeian DAG and the OVO SVM.

The results obtained for the associated run-time experiments are presented in Table 4.15. Again the table presents both the generation and classification times. From the table it can be observed, that the lowest generation time was recorded when using the OVO SVM classification model. While the lowest classification time was recorded when using the Naive DAG classification model. However, although the presented generation times show that OVO SVM requires less time to be generated, the presented generation times for OVO SVM do not consider the time required for searching for the optimal values for the SVM parameters C and γ.

Regarding the comparison between stand-alone CARM, Bagging of CARM, and the CARM based DAG model the results are presented in Table 4.16 in terms of average accuracy and average AUC (best results highlighted in bold font). From the table it can be observed that the rooted DAG classification model improves classification accuracy with respect to six of the eleven datasets considered (WaveForm, Wine, Nursery,

TABLE 4.14: Average Accuracy and AUC values obtained using Naive Bayes DAG coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier

Data set	Classes	Naive Bayes DAG		OVO SVM	
		Acc.	AUC	Acc.	AUC
WaveForm	3	77.00	0.77	**80.72**	**0.81**
Wine	3	**95.08**	**0.95**	93.13	0.93
Nursery	5	90.28	0.45	**99.69**	**0.64**
heart	5	**55.57**	**0.35**	53.01	0.22
PageBlocks	5	**92.69**	**0.52**	92.58	0.50
Dermatology	6	87.23	0.85	**88.73**	**0.86**
glass	7	**72.99**	**0.51**	72.04	0.47
Zoo	7	93.18	**0.59**	**94.00**	0.58
ecoli	8	**84.43**	**0.41**	82.95	0.36
led	10	75.56	**0.76**	**75.62**	**0.76**
PenDigits	10	83.58	0.83	**98.60**	**0.99**
soybean	15	**90.75**	**0.92**	92.54	0.91
Mean		83.20	0.66	**85.30**	**0.67**

TABLE 4.15: Run time results (in seconds) obtained using Naive Bayes DAG coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier

Data set	Generation Time		Classification Time	
	DAG	OVO	DAG	OVO
WaveForm	0.199	1.183	0.009	0.385
Wine	0.189	0.066	0.009	0.029
Nursery	5.982	10.974	0.595	0.778
heart	0.333	0.139	0.015	0.067
PageBlocks	2.510	0.657	0.266	0.135
Dermatology	0.445	0.201	0.020	0.054
glass	0.539	0.120	0.016	0.044
Zoo	0.491	0.066	0.009	0.028
ecoli	1.032	0.101	0.031	0.038
led	33.701	0.582	0.261	0.182
PenDigits	264.369	4.153	0.723	0.529
soybean	1520.706	0.386	0.068	0.151
Mean	152.541	**1.552**	**0.169**	0.202

Heart, Dermatology, and Ecoli). For another four datasets (PageBlocks, Glass, Zoo, and Led) the stand-alone CARM produced the best result although for one dataset (Led) the same result was produced as in the case of bagging. For one dataset (PenDigits) bagging classification produced the best result. With respect to the statistical evaluation, as in the case of Naive Bayesian DAG and Decision Tree DAG, no statistically significant difference was detected in the operation of the proposed CARM DAG and

the conventional classification methods.

The results obtained for the run-time experiments are presented in Table 4.17. As before the table presents both the generation and classification times. From the table it can be observed, and again as expected, that the lowest generation and classification times were obtained when using stand-alone CARM.

TABLE 4.16: Average accuracy and AUC results obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM classifiers and (iii) DAG classification with CARM classifiers at nodes

Data set	Classes	CARM		Bagging		DAG	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	60.04	0.60	60.76	0.61	**68.54**	**0.69**
Wine	3	71.88	0.74	61.48	0.61	**86.26**	**0.86**
Nursery	5	73.94	0.36	73.94	0.36	**86.81**	**0.43**
Heart	5	51.70	0.20	45.49	0.24	**53.76**	**0.20**
PageBlocks	5	**89.99**	**0.21**	89.95	0.21	89.77	0.20
Dermatology	6	77.00	0.66	72.12	0.62	**80.82**	**0.77**
Glass	7	**65.05**	**0.43**	53.30	0.31	61.71	0.39
Zoo	7	**94.00**	**0.59**	83.00	0.46	88.00	0.52
Ecoli	8	**49.98**	0.12	37.90	0.07	39.51	**0.28**
Led	10	**67.28**	**0.67**	67.09	**0.67**	40.06	0.43
PenDigits	10	75.99	0.76	**77.09**	**0.77**	46.76	0.47
Mean		**70.62**	**0.49**	65.65	0.45	67.45	0.48

TABLE 4.17: Run time results (in seconds) obtained when using: (i) stand alone CARM classification, (ii) bagging of CARM classifiers and (iii) DAG classification with CARM classifiers at nodes

Data set	Classes	Generation Time			Classification Time		
		CARM	Bagging	DAG	CARM	Bagging	DAG
Waveform	3	1.254	1.418	2.494	0.002	0.004	0.234
Wine	3	0.389	0.559	0.682	0.000	0.000	0.014
Nursery	5	1.142	1.222	5.547	0.003	0.004	0.594
Heart	5	0.383	0.624	3.788	0.000	0.002	0.015
PageBlocks	5	0.796	0.890	3.404	0.001	0.001	0.257
Dermatology	6	0.451	0.660	8.542	0.000	0.001	0.023
Glass	7	0.279	0.316	2.281	0.001	0.001	0.015
Zoo	7	0.312	0.549	14.938	0.000	0.000	0.006
Ecoli	8	0.238	0.276	0.992	0.000	0.000	0.023
Led	10	0.581	0.647	25.451	0.001	0.015	0.263
PenDigits	10	2.199	2.578	2402.572	0.008	0.010	1.195
Mean		**0.729**	0.885	224.608	**0.001**	0.003	0.240

4.5 Summary

A hierarchical ensemble classification model for multi-class classification based on a *rooted* Directed Acyclic Graph (DAG) structure has been presented in this chapter. Three different classification algorithms were used to generate node classifiers: (i) decision tree, (ii) Naive Bayes and (iii) CARM. The *rooted* DAG structure facilitated the use of two mechanisms to address the successive mis-classification problem associated with hierarchical classifiers where a mis-classification near the root of the hierarchy is passed on down the hierarchy. The first proposed mechanism was the combination technique for grouping classes across nodes at individual levels in the DAG so that an overlap existed between the class groups. The second mechanism was the option to follow multiple paths down the hierarchy by utilising the probability or confidence values generated by Naive Bayes and CARM classifiers respectively.

The operation of the proposed *rooted* DAG model was compared with three well-established more conventional classification models: (i) stand-alone classification, (ii) Bagging ensemble classification, and (iii) OVO classification. More specifically, the performance of each type of *rooted* DAG model considered (decision tree, Naive Bayes and CARM) was compared with the same stand alone classifier that was used in the context of DAG and also with bagging ensemble classification using same classification algorithms (so that a "consistent comparison" was obtained). In addition, a "non-consistent" comparison between Naive Bayes DAG, because the Naive Bayes DAG produces the best performance, and OVO SVM was conducted. The objective of this last comparison was to compare the operation of the proposed *rooted* DAG model with one of the state of the art methods for multi-class classification.

The reported evaluation demonstrated that best results were produced when using the Naive Bayes algorithm to generate the classifiers at the nodes within the *rooted* DAG compared to the decision tree and CARM based DAG models. With respect to following multiple paths within the *rooted* DAG, unlike in the case of the Binary Tree hierarchical classification model, following multiple paths within the DAG classification model was found to be not significantly more effective than when following only a single path. The reason for this was argued to be that the combination techniques used to distribute classes between nodes resulted in well-defined class labels at each DAG node, unlike the clustering algorithms that were used with respect to the Binary Tree model; consequently the mis-classification was less and the effect of following multiple paths within the DAG was not highly significant. Any number of branches may be explored at each DAG node, however it was suggested that it is preferable to explore a maximum of two branches at each node because of efficiency issues. The maximum number of branches to be explored can of course be considered as a user specified value. Although the proposed *rooted* DAG classification model improved the classification effectiveness with respect to some of the considered data sets, no statistically significant difference was detected in the operation of the proposed *rooted* DAG and the conventional classification methods.

The issues with the *rooted* DAG classification model are: (i) as the number of class labels featured in the dataset increases the number of classifiers that need to be generated increases correspondingly, as a result more storage and run time are required to generate the model (thus there are scalability and efficiency issues); and (ii) the combination mechanism and the Multiple Path strategy only partly mitigates against the early misclassification issue (effectiveness issue).

In order to improve the performance (scalability, effectiveness and efficiency) of the *rooted* DAG model, a *non-rooted* DAG structure, rather than a *rooted* DAG structure, was considered. The advantageous features provided by the *non-rooted* DAG structure are:

1. It enables the elimination of the root node where the largest number of class combinations are considered.

2. As a result of (1) it has the effect of reducing the overall number of levels in the desired model (depth pruning).

3. It enables the application of *breadth pruning*, thus allowing for the elimination of "weak" classifiers at each DAG level, so as to reducing the overall size of the DAG further. Note that breadth pruning cannot be applied in the case of the *rooted* DAG structure because the *rooted* DAG requires inclusion of all classes combinations.

The *non-rooted* DAG is considered in the next chapter.

Chapter 5

Directed Acyclic Graph (DAG) Structure Based Hierarchical Classification Model

5.1 Introduction

This chapter considers using a *non-rooted* Directed Acyclic Graph (DAG) structure, rather than a *rooted* DAG structure, to generate a hierarchical classification model. Recall from the previous chapter that a *rooted* DAG, although straightforward to generate, entails a number of disadvantages in the context of: (i) scalability, (ii) effectiveness and (iii) efficiency. The proposed *non-rooted* structure seeks to address the disadvantages of the *rooted* DAG. The advantages offered by the *non-rooted* DAG structure are:

1. It enables the elimination of the root node where the largest number of class combinations are considered, as well as reducing the overall number of levels in the desired model (depth pruning).

2. It enables the application of breadth pruning, reducing the number of classifiers that need to be generated at each DAG level so as to reduce the overall size of the DAG further.

Note that breadth pruning is not applicable to the *rooted* DAG model because the *rooted* DAG requires inclusion of all classes combinations. For ease of understanding, this chapter considers the *non-rooted* DAG without breadth pruning, while chapter 6 considers the *non-rooted* DAG with breadth pruning.

The *non-rooted* DAG hierarchical classification model is another form of ensemble classifier of the form promoted in this thesis. Each node in the DAG holds a classifier. The classifiers at the leaves conduct fine-grained classifications while the classifiers at non-leaf nodes conduct coarse-grained classification directed at classifying examples using groups of labels (as in the previous cases considered). In order to group (partition) the input data D during the hierarchy generation process the combinations techniques

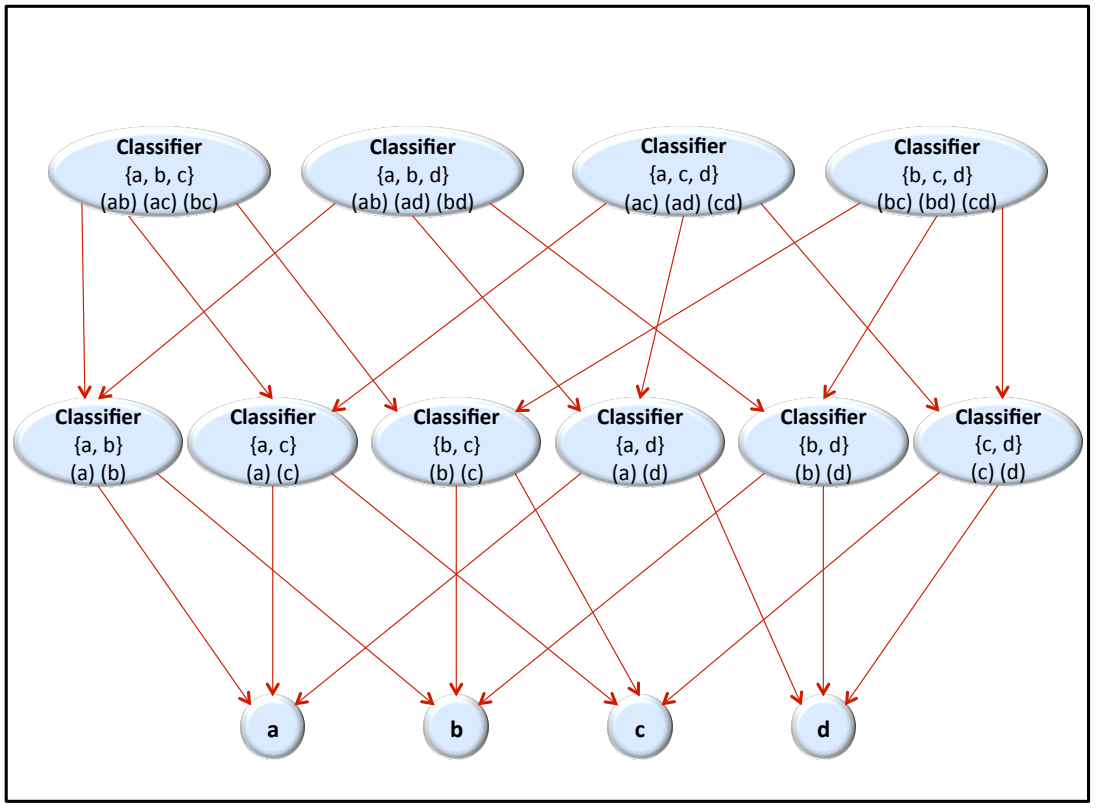


FIGURE 5.1: DAG example.

proposed in Chapter 4 will again be used. A simple example DAG classifier for four class labels, C $= \{a, b, c, d\}$, is presented in Figure 5.1. The first level nodes are assigned class combinations of size three ($|C| - 1$), while the second level nodes are assigned class combinations of size two ($|C| - 2$). The distinction between this DAG structure and the *rooted* DAG structure can clearly be seen by comparison with Figure 4.1.

An issue with respect to the *non-rooted* DAG structure, as the name implies, is the need to determine the "starting node" (a root) from which the classification process is to commence. To this end classifier generators, such as Naive Bayesian Classification or Classification Association Rule Miners (CARM) were again used to produce probability or confidence values that can be utilised to determine the starting node. As mentioned earlier, successive mis-classification is the overriding general drawback of hierarchical classification, whereby if an example is mis-classified early on in the process (near the root of the DAG) it will continue to be mis-classified at deeper levels, regardless of the classifications proposed at lower level nodes and the final leaf nodes. Again, to address this problem a Multiple Path strategy is proposed (facilitated by the probability or confidence values generated by Naive Bayes classifiers or CARM hosted at the DAG nodes).

The rest of this chapter is organised as follows: Section 5.2 explains the generation of the DAG hierarchical ensemble classification model in detail. While Section 5.3 considers

the operation of the proposed model. Section 5.4 presents the conducted experiments and the obtained results. Finally, a summary of the chapter is presented in Section 5.5.

5.2 Non-Rooted DAG Generation

In this section the generation of the *non-rooted* DAG hierarchical ensemble classification model is described. The process requires that a classifier is generated for each DAG node using an appropriate training set, which ideally includes well-defined class label groupings (individual classes at leaf nodes). As noted earlier groupings are identified using a combination mechanism. At each level in the DAG the class groupings are determined by finding all possible classes combinations of size $|C| - i$ (where i is the level counter, at start up $i = 1$). As the process proceeds i is increased by one, as a result the combination (group) size is decreased by one. The process terminates when the combination size becomes two.

Algorithm 11 presents the generation process in more detail. The input to the algorithm is the training data set D and the set of class labels C. The DAG is created in a recursive manner using the function *dagGen*. On each recursion the *dagGen* function is invoked with two parameters: k, the combination size (starting with $k = |C| - 1$ and ending with $k = 2$); and $CurrentNodes$, a reference to the current level nodes (resulting from the previous iteration, initially $CurrentNodes = null$). The recursive process starts by finding the set of size k class combinations, the set C_k (line 13). After that we go through the set C_k (line 15) and on each iteration: (i) the set of training set examples T_i that feature the combination $C_i \in C_k$ is identified (line 16), (ii) a classifier G_i using T_i is trained (line 17); (iii) a new DAG node, *node*, is created (line 18); and (iv) the new node is added to the set of accumulated level k nodes so far, $NodeSet$ (line 19). Then, if the current level is not the first level in the DAG (Line 20), we loop through the set of current nodes and add a link from each current node $CurrentNode_j$ to the new node *node* whenever the set of class labels associated with the new node (C_i) is included in the set of class labels associated with a current node ($C_i \subset CurrentNodes_j.C$). The recursive process terminates if k reaches 2 (line 28).

The conjectured advantage of using a *non-rooted* DAG structure to generate the desired DAG classification model is that it could result in better classification accuracy because of: (i) the elimination of the root node that distinguishes between a large number of class combinations, and (ii) the flexibility of the *non-rooted* DAG structure, and the combination mechanism, that enables the reduction of the overall number of levels in the desired model. More specifically, this flexibility allows the generation of DAG according to any predefined number of levels. In the conducted experiments generating the DAG with various numbers of levels was considered, the objective was to gradually observe the effect of reducing the number of levels on the classification performance. For example for generating all levels in the DAG, the DAG model is created exactly as described in algorithm 11 (the combination sizes range from $|C|-1$ to 2) and the number of classifiers

Algorithm 11 DAG Generation

```
1: INPUT
2: D = The input training dataset
3: C = The set of Classes featured in D
4: OUTPUT
5: The generated DAG
6:
7: Start
8: k = |C| − 1
9: dagGen(k, null)
10: End
11:
12: function dagGen(k, CurrentNodes)
13:     C_k = Set of size k combinations in C
14:     NodeSet = {}
15:     for i = 1 to i = |C_k| do
16:         T_i = Set of training examples in D that feature C_i (T_i ⊂ D)
17:         G_i = Classifier for C_i built using training set T_i
18:         node = new Node(G_i, C_i)
19:         NodeSet = NodeSet ∪ node
20:         if CurrentNodes != Null then
21:             for j = 1 to j = |CurrentNodes| do
22:                 if C_i ⊂ CurrentNodes_j.C then
23:                     CurrentNodes_j.childNodes = CurrentNodes_j.childNodes∪node
24:                 end if
25:             end for
26:         end if
27:     end for
28:     if k > 2 then
29:         dagGen(k − 1, NodeSet)
30:     end if
31: end function
```

required to be trained in order to generate all levels in the DAG classification model can be determined using Equation 5.1:

$$NumberOfClassifiers = 2^N - N - 2 \tag{5.1}$$

where N is the number of class labels in a given dataset.

While if only two levels needed to be generated, for example, the permitted combination sizes range from 3 to 2, the number of classifiers that need to be generated in this case can be calculated as follows (Equation 5.2):

$$NumberOfClassifiers = (1/6)\,(N)\left(N^2 - 1\right) \tag{5.2}$$

where N is the number of class labels in a given dataset.

The conjecture here is that by reducing the number of levels in the DAG the classification performance (with respect to efficiency, effectiveness, and scalability) of the DAG

classification model might be enhanced because: (i) the number of classifiers that need to be generated will be reduced and as a result the proposed model can be generated for datasets that feature larger number of class labels than in the case of *rooted* DAG, (ii) the internal classifiers are not required to discriminate between large numbers of class combinations and (iii) the number of classifiers that need to be evaluated during the classification stage will be decreased, as a result the probability of mis-classification will also be decreased.

5.3 Non-Rooted DAG Operation

In this section the operation of the *non-rooted* DAG hierarchical ensemble classification model is explained. Two methods of operation were considered: (i) the Single Path strategy and (ii) the Multiple Path strategy. A challenging issue associated with both strategies is how to identify the best starting node among the set of nodes at the first level in a given DAG. In both cases this is addressed by using the probability values associated with the Naive Bayes classifiers generated for each DAG node, or the confidence values generated by CARM. A disadvantage of the Single Path strategy is that it is susceptible to the successive mis-classification issue discussed earlier. The Multiple Path strategy seeks to address this issue by again using the probability values associated with the Naive Bayes classifiers (or confidence values associated with CARM) to decide, at each node, whether to follow single or multiple paths. The two strategies are discussed in the following two subsections.

5.3.1 Single Path Strategy

As the name suggests, in the Single Path strategy only a "Single path" will be followed according to the classification at each DAG node. The Single Path classification strategy can be viewed as a two-step process: (i) determine a best start node amongst the set of nodes available at the first level in the DAG by evaluating all the classifiers that exist at the first level and selecting the node with the classifier that generates the highest probability value (or confidence value if using CARM) and then (ii) drilling down as dictated by subsequent internal node classifications until a classifier that can assign a single class label to the given example is arrived at.

Algorithm 12 presents the Single Path procedure. Note that the algorithm assumes the use of Naive Bayes classifiers at each DAG node; the procedure is the same when using CARM, but instead of using probability values the associated confidence values are used. The input to the algorithm are: (i) e, a new unseen example; and (ii) a reference to the nodes at the first level in the given DAG $FirstLevelNodes$ (from which all the DAG child nodes can be identified). The output is a predicted class label for e. The process commences by identifying the best starting node among nodes at the first level in the DAG (line 8-15) by looping through the nodes at the first level (line 9) and for each node: classifying e using the respective node classifier (line 10), and adding the resulting

class group with the associated probability to S, the set of class groups and associated probabilities resulting from evaluating first level nodes (line 12). The best start node is then the node with the highest associated probability value (line 14). The next node will be the child node of the identified $startNode$ representing class group C_i associated with $max(Pi)$ (Line 15). The next step is a recursive process using the $dagclassify$ function described on lines 19 to 27. The $dagclassify$ function is called with two parameters: e, the example to be classified, and $Node$ a pointer to the current node location in the DAG. On each recursion the example e is classified using the classifier at the current DAG node, $Node : Gi$ (line 20). The process proceeds depending on the nature of the returned class label. If it is a single class label then we return this class (line 22). If we have a group of class labels, $dagClassify$ is called again (line 25) with e and a pointer ($ChildNode$) to the child node associated with the identified class group.

Algorithm 12 DAG Single Path Classification

```
1: INPUT
2: e = A new unseen example
3: FirstLevelNodes = nodes at the first level in the DAG
4: OUTPUT
5: The predicted class label c for the input example e
6:
7: Start
8: S = Classification results for e using the classifiers at the first level in the DAG comprised
   of: (i) class groups and (ii) the associated Bayesian probability values (initially S = {})
9: for j = 1 to j = |FirstLevelNodes| do
10:     Ci =Classification result for e using classifier Node.Gi
11:     Pi = Bayesian probability value associated with class group Ci
12:     S = S ∪ Ci associated with Pi
13: end for
14: startNode = node associated with max(Pi) in S
15: ChildNode = child node for startNode representing class group Ci associated with max(Pi)
16: c = dagclassify(e, ChildNode)
17: End
18:
19: function dagClassify(e, Node)
20:     C = Classification result for e using classifier Node.G_i
21:     if |C| == 1 then
22:         return c (c ∈ C)
23:     else
24:         ChildNode = child node representing class group C
25:         return (dagclassify(e, ChildNode))
26:     end if
27: end function
```

It is interesting to note that it might be the case that several classifiers at the first level generate exactly the same probability or confidence value (this phenomena often occurs when using CARM because of the generality of the rules produced), the problem here is if these nodes refer to different nodes in the next level. To handle this situation a mechanism whereby an exception was evaluated causing the system to proceed to the next level node, which has had the highest number of links from previous level nodes

(majority voting). If there are more than one such node, then the next node is chosen at random.

The number of classifiers that need to be used to classify an example using the Single Path strategy, is all the classifiers at the first level and a classifier at each subsequent DAG level. Equation 5.3 can be used to determine the number of classifiers needed to classify an example using the Single Path strategy when considering DAG with all levels, where N is the number of class labels in a given dataset. If we have only a two level DAG then the number of classifiers to be evaluated is determined using Equation 5.4.

$$NumberOfClassifiers = 2N - 3 \tag{5.3}$$

$$NumberOfClassifiers = (1/6)\,(N-2)\,(N-1)\,(N) + 1 \tag{5.4}$$

5.3.2 Multiple Path Strategy

As before, the Multiple Path strategy is designed to address the successive mis-classification issue, discussed earlier, associated with hierarchical classification. In the Multiple Path strategy more than one path can be followed within the DAG classification model. Although many branches can be followed at each DAG node, only two branches are suggested as a maximum because of the efficiency issue discussed earlier in Chapter 4. A second reason is that for evaluation purposes comparisons can be made with the binary tree hierarchical ensemble model (where only a maximum of two paths can be followed at each tree node). As mentioned earlier, Naive Bayes classifiers and Classification Association Rule Miners (CARM) were used, so the Bayesian probability p (or confidence value) associated with the individual class groups, at each DAG node, will be used to dictate whether one or two branches will be followed according to the predefined threshold σ. Where at each DAG node the two class groups associated with the highest probabilities (highest confidence) are identified, then if their probabilities (confidence values) are greater than σ both branches will be explored, otherwise the branch with the highest associated probability (confidence) value will be selected.

In order to decide the final class label from the collection of "candidate classes" resulting from following multiple paths, the accumulated weight scheme was adopted because the previous chapters concluded that this scheme tended to produce a better classification performance. Using the accumulated weight scheme we take into consideration all probability (or confidence) values in a followed path to produce an accumulated value. More specifically, the probability (or confidence) values for a followed path are added and then divided by the number of classifiers used in the path to produce a $NormalisedAccumulated\ Probability$ (or $NormalisedAccumulatedConfidence$) value, ($0 < NormalisedAccumulatedProbability < 1$, while $0 < NormalisedAccumulatedConfidence \leq 100$). The normalised accumulated probability (or confidence) value is calculated for each candidate class, the candidate class associated with the highest value will be retrieved as the class label for a given example.

The Multiple Path classification strategy can be viewed as a three-step process: (i) determining the start node(s) from the set of nodes available at the first level in the DAG by evaluating all the classifiers that exist at this first level and selecting one or two nodes as start nodes based on the probability threshold σ (confidence threshold in case of using CARM), (ii) for each identified node drill down following one or two paths as indicated and repeat until a classifier that can assign a single class label to the given example is arrived at and finally (iii) identify the class label associated with the highest generated accumulated weight value.

Algorithm 13 (a and b) summarises the multiple path procedure. Note here that the algorithm assumes usage of Naive Bayes classifiers at each DAG node, the procedure is the same when using CARM, but instead of using probability values the associated confidence values are used. The inputs to the algorithm are: (i) the new unseen example e; (ii) a reference, $FirstLevelNodes$, to the first level DAG; and (iii) the path selection threshold σ. For simplicity the algorithm is decomposed into two main functions: $dagFirstLevelMulti$, and $dagMultiPathClassify$.

Starting with $dagFirstLevelMultiPathClassify$ (Algorithm 13 (a)), which is responsible for determining the start node (or nodes) amongst the set of nodes available at the first level. The process commences by evaluating all the classifiers that exist at the first level (lines 20-24), and selecting the two nodes that generate the highest probability values (lines 25-28). If the second highest probability value is greater than σ then both nodes will be considered as start nodes, otherwise only the node that generated the highest probability value will be considered as the start node (lines 29-32).

After determining the start node(s) the recursive function $dagMultiPathClassify$ is called (Algorithm 13 (b)). The $dagMultiPathClassify$ function operates in a similar manner to the $dagClassify$ function presented in Algorithm 12 except that it: (i) uses the σ threshold to decide whether one or two branches will be followed, (ii) uses the variable *accumProb* to store the accumulated Bayesian probability values in a followed path, (iii) maintains a *counter* to count the number of probability values in a followed path, and (iv) uses a data structure $Path$, in which to hold candidate class labels with their associated normalised Bayesian probability values. On each recursion of the $dagMultiPathClassify$ function the Bayesian classifier held at the current node is used to produce a probability value with respect to e for each class group. Only the class groups associated with the two highest probability values are considered (two branches will be followed at maximum) (lines 35-40). Whenever the size of a class group considered at a node is equal to one (Lines 41 and 49), indicating that the group comprises a single class label, the class label and associated normalised probability value are added to $Path$ (lines 43 and 51). Note that the normalised probability is calculated by dividing the accumulated probability generated so far *accumProb*, by the number of classifiers used in the current path *counter* (lines 42 and 50). Whether one or two branches are followed, at each DAG node, depends on the probability values returned using the Bayesian classifier at the current node and the σ threshold. If the second

highest probability value is greater than σ (line 48) then two branches will be followed, otherwise only a single branch will be followed. At the end of the process the $Path$ data structure is processed to identify the class label with the highest associated normalised probability value (line 14).

Algorithm 13 (a) DAG Multiple Path Classification

```
 1: INPUT
 2: e = A new unseen example
 3: FirstLevelNodes = nodes at the first level in the DAG
 4: σ = Path selection threshold
 5: OUTPUT
 6: The predicted class label c for the input example e
 7:
 8: GLOBAL VARIABLES
 9: Path = {} (Set of identified paths each comprised of: (i) a class label and
10:          (ii) an associated normalised Bayesian probability value)
11:
12: Start
13: dagFirstLevelMultiPathClassify(e, FirstLevelNodes)
14: c = Class label with highest probability value in Path
15: End
16:
17: function dagFirstLevelMultiPathClassify(e, FirstLevelNodes)
18:     S = Classification results for e using the classifiers at the first level in the DAG comprised
19:      of: (i) class groups and (ii) the associated Bayesian probability values (initially S = {})
20:     for j = 1 to j = |FirstLevelNodes| do
21:         Ci =Classification result for e using classifier Node.Gi
22:         Pi = Bayesian probability value associated with class group Ci
23:         S = C ∪ Ci with associated probability Pi
24:     end for
25:     p1 = the highest probability in S (max(Pi) in S)
26:     p2= the second highest probability in S
27:     startNode1 = node associated with p1
28:     startNode2 = node associated with p2
29:     dagMultiPathClassify(e, startNode1, 0, 0)
30:     if p2 ≥ σ then
31:         dagMultiPathClassify(e, startNode2, 0, 0)
32:     end if
33: end function
```

In the worst case the number of classifiers needed to classify an example using the Multiple Path strategy (considering all levels in the DAG) is given by Equation 5.5. If we have only two levels DAG then the number of classifiers to be evaluated (in the worst case) is as given by Equation 5.6.

$$NumberOfClassifiers = N + 2^{N-1} - 4 \tag{5.5}$$

$$NumberOfClassifiers = (1/6)\,(N-2)\,(N-1)\,(N) + 4 \tag{5.6}$$

where N is the number of class labels in a given dataset.

Algorithm 13 (b) DAG Multiple Path Classification

```
34: function dagMultiPathClassify(e, Node, accumProb, counter)
35:     C = Classification result for e using classifier Node.G_i
36:     P = Bayesian probability values associated with each class group in C
37:     C_1 = Class group in C associated with highest probability value
38:     p_1 = Bayesian probability associated with C_1
39:     C_2 = Class group in C associated with second highest probability value
40:     p_2 = Bayesian probability associated with C_2
41:     if |C_1| == 1 then
42:         normProb = (AccumProb + p_1)/(counter + 1)
43:         Path = Path ∪ ⟨c, normProb⟩ (c ∈ C_1)
44:     else
45:         ChildNode = child node representing class group C_1
46:         dagMultiPathClassify(e, ChildNode, accumProb + p_1, counter + 1)
47:     end if
48:     if p_2 ≥ σ then
49:         if |C_2| == 1 then
50:             normProb = (AccumProb + p_2)/(countert + 1)
51:             Path = Path ∪ ⟨c, normProb⟩ (c ∈ C_2)
52:         else
53:             ChildNode = child node representing class group C_2
54:             dagMultiPathClassify(e, ChildNode, accumProb + p_2, counter + 1)
55:         end if
56:     end if
57: end function
```

5.4 Experiments and Results

This section presents an overview of the adopted experimental set up and the evaluation results obtained. The effectiveness of the DAG classification model was evaluated using twelve different data sets (with various numbers of class labels) taken from the UCI machine learning repository [63], and pre-processed using the LUCS-KDD-DN software [23] as described in Chapter 1. Note here that the WaveForm and Wine datasets were not considered in the evaluation, because these datasets feature three class labels and consequently the minimum required two DAG levels can not be generated for these datasets. Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were again average accuracy and average AUC (Area Under the receiver operating Curve). As in the case of the evaluation sections presented in Chapters 3 and 4, although the results in terms of average accuracy and average AUC are both included in this section, we will discuss the results only in terms of average AUC (because of the theoretical and empirical evidences that AUC is a better measure than accuracy in evaluating learning algorithms [52]). The results obtained are presented in the following sections as follows: Section 5.4.1 presents the results obtained using the Single Path strategy. Section 5.4.2 presents the results obtained using the Multiple Path strategy. Section 5.4.3 provides a comparison between the DAG Single Path and Multiple Path

strategies. Section 5.4.4 presents a comparison between *rooted* and *non-rooted* DAGs for hierarchical classification.

5.4.1 Single Path Experiments and Results

This section presents the results obtained using the Single Path strategy with respect to the two alternative classification algorithms, Naive Bayes and CARM, considered to generate the DAG ensemble classification model. The objective was to identify the most effective and efficient classifier to generate a DAG ensemble classification model.

Starting with the results obtained when using Naive Bayesian DAG. As mentioned earlier, because of the flexibility of the DAG structure, we can generate the DAG with any predefined number of levels. For each dataset experiments were conducted using different numbers of levels in the DAG, starting from the maximum number of $N-2$ levels (where N is the number of class labels in the dataset) to two levels. Table 5.1 presents the results obtained. In the table best results are highlighted in bold font, results with a gray background indicate that the DAG with the specified number of levels does not exist and thus the presented result in that cell is the result when using the maximum number of levels for that dataset. From the table it can be observed that there is little difference between the obtained results. Although the Two-level DAG (the minimum number of DAG levels) generated the best overall results in terms of average AUC, according to the conducted statistical evaluation, this difference in performance is not statistically significant. However, an advantage of the Two-level DAG is that it can be applied to datasets that feature larger numbers of class labels, such as Chess KRvK and Letter Recognition, that was not possible using *rooted* DAG structure. The reader might think about generating only one level, however this would not be a DAG, it will be more of a kind of OVO strategy whereby classifiers are generated for each of the possible combinations of size two and a voting scheme is used in the classification stage. However, additional experiments were conducted whereby a OVO strategy (using Naive Bayes) was applied to the datasets; the obtained results were exactly the same as when using a stand-alone Naive Bayes classifier (Naive Bayes is a multi-class classifier).

With respect to using CARM to generate the classifiers at DAG nodes, two variations of the DAG were considered: generating the maximum number of levels (thus $N-2$ levels) and generating the minimum number of levels (two levels) with respect to each dataset. Table 5.2 presents the obtained results for the CARM DAGs coupled with the Single Path strategy. From the table it is interesting to note that as the number of DAG levels is reduced the recorded classification accuracy is also reduced. More specifically, the $N-2$ levels DAG produced a better classification accuracy than the two levels DAG.

TABLE 5.1: Average Accuracy and AUC values obtained using the Naive Bayes DAG model coupled with the Single Path strategy when using different numbers of levels in the DAG

Data set	Classes	Two levels		Three levels		Four levels		Five levels		Six levels		All levels	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Nursery	5	58.03	**0.30**	56.93	0.29	56.93	0.29	56.93	0.29	56.93	0.29	56.93	0.29
Heart	5	54.19	**0.35**	55.22	**0.35**	55.22	**0.35**	55.22	**0.35**	55.22	**0.35**	55.22	**0.35**
PageBlocks	5	91.83	**0.53**	91.83	**0.53**	91.83	**0.53**	91.83	**0.53**	91.83	**0.53**	91.83	**0.53**
Dermatology	6	86.66	**0.85**	87.23	**0.85**	87.23	**0.85**	87.23	**0.85**	87.23	**0.85**	87.23	**0.85**
Glass	7	59.49	**0.49**	63.70	0.44	67.90	0.45	69.81	0.46	69.81	0.46	69.81	0.46
Zoo	7	94.18	**0.61**	93.18	0.59	92.18	0.58	92.18	0.58	92.18	0.58	92.18	0.58
Ecoli	8	80.23	0.37	83.26	**0.41**	83.22	0.40	84.43	**0.41**	84.43	**0.41**	84.43	**0.41**
Led	10	75.56	0.75	75.66	**0.76**	75.66	**0.76**	75.66	**0.76**	75.66	**0.76**	75.66	**0.76**
PenDigits	10	83.62	**0.84**	83.44	0.83	83.57	0.83	83.58	0.83	83.58	0.83	83.58	0.83
Soybean	15	90.39	**0.92**	90.57	**0.92**	90.57	**0.92**	90.75	**0.92**	90.75	**0.92**	90.75	**0.92**
Mean		77.418	**0.601**	78.102	0.597	78.431	0.596	78.762	0.598	78.762	0.598	78.762	0.598

TABLE 5.2: Average Accuracy and AUC values obtained using the CARM DAG model coupled with the Single Path strategy when using different numbers of levels in the DAG

Data set	Classes	All-Level DAG		Two-level DAG	
		Acc.	AUC	Acc.	AUC
Nursery	5	**86.81**	**0.43**	**86.81**	**0.43**
Heart	5	**53.07**	**0.20**	51.70	**0.20**
PageBlocks	5	91.27	**0.40**	**91.47**	**0.45**
Dermatology	6	**79.62**	**0.71**	65.83	0.55
Glass	7	**61.71**	**0.36**	59.81	0.34
Zoo	7	**88.00**	**0.52**	86.00	0.50
Ecoli	8	**32.42**	**0.25**	62.94	0.23
Led	10	**42.06**	**0.43**	19.28	0.18
PenDigits	10	**41.62**	**0.42**	18.95	0.19
Mean		**64.06**	**0.41**	60.31	0.34

The reason for this is related to the issue, discussed earlier, whereby several classifiers at the first level generate exactly the same confidence value, because of the generality of the rules, but the nodes where these classifiers are held link to different nodes at the next level. This issue is of greater significance with respect to the two levels DAG because the number of nodes that exist at the first level in this case is greater than the number of nodes at the first level in case of the $N-2$ level DAG (All-level DAG).

Although there was an observed difference in the effectiveness between All-level and Two-level CARM DAGs, this difference was not found to be statistically significant.

Table 5.3 presents a comparison between the operation of the Naive Bayes DAG and CARM DAG when coupled with the Single Path strategy, in terms of average accuracy and average AUC. From the table it can be clearly observed that when using a Naive Bayes classifier to generate the DAG, regardless of the number of levels in the DAG, produced the best results with respect to the the majority of the considered datasets. Naive Bayes DAGs significantly outperforms CARM DAGs.

The results obtained for the run-time experiments, with respect to both the Naive Bayes DAG and CARM DAG, when using the Single Path strategy are presented in Table 5.4. The table presents the generation and classification times in each case. From the table it can be observed that the lowest generation and classification times were obtained when using Naive Bayes to generate the DAG ensemble classification model.

In conclusion, the results presented in this section corroborates the results obtained when using *rooted* DAG as presented in the previous chapter (Chapter 4) where Naive Bayes classification was also found to be the most effective and efficient classifier with which to generate a DAG ensemble classification model. With respect to number of levels in the Naive Bayes DAGs, although the Two-level DAG (the minimum number of DAG levels) generated the best overall results in terms of average AUC, according to the conducted statistical evaluation, this difference in performance was not statistically significant. However, an advantage of the Two-level DAG is that it can be applied to

TABLE 5.3: Comparison of average Accuracy and AUC values obtained using Naive Bayes DAGs and CARM DAGs coupled with the Single Path strategy

Data set	Naive DAGs				CARM DAGs			
	All-level		Two-level		All-level		Two-level	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	56.93	0.29	58.03	0.30	**86.81**	**0.43**	**86.81**	**0.43**
Heart	**55.22**	**0.35**	54.19	**0.35**	53.07	0.20	51.70	0.20
PageBlocks	**91.83**	**0.53**	**91.83**	**0.53**	91.27	0.22	91.47	0.45
Dermatology	**87.23**	**0.85**	86.66	**0.85**	79.62	0.71	65.83	0.55
Glass	**69.81**	0.46	59.49	**0.49**	61.71	0.36	59.81	0.34
Zoo	92.18	0.58	**94.18**	**0.61**	88.00	0.52	86.00	0.50
Ecoli	**84.43**	**0.41**	80.23	0.37	32.42	0.25	62.94	0.23
Led	**75.66**	**0.76**	75.56	0.75	42.06	0.43	19.28	0.18
PenDigits	83.58	0.83	**83.62**	**0.84**	41.62	0.42	18.95	0.19
Mean	77.43	0.56	**75.98**	**0.57**	64.06	0.39	60.31	0.34

TABLE 5.4: Run time results (in seconds) obtained using Naive Bayes DAGs and CARM DAGs and the Single Path strategy

Data set	Generation Time				Classification Time			
	Naive		CARM		Naive		CARM	
	All-level	Two-level	All-level	Two-level	All-level	Two-level	All-level	Two-level
Nursery	5.011	3.275	5.607	3.484	0.017	0.021	0.040	0.028
Heart	0.261	0.262	3.243	2.385	0.001	0.001	0.001	0.001
PageBlocks	2.131	1.554	3.736	2.568	0.014	0.013	0.010	0.018
Dermatology	0.391	0.310	7.010	4.048	0.001	0.003	0.002	0.004
Glass	0.490	0.290	2.650	1.341	0.001	0.003	0.001	0.004
Zoo	0.437	0.262	12.259	5.103	0.000	0.225	0.000	0.003
Ecoli	0.944	0.373	1.746	0.592	0.002	0.006	0.001	0.009
Led	31.803	1.095	42.367	1.503	0.013	0.034	0.029	0.057
PenDigits	244.986	4.663	2593.424	88.918	0.051	0.069	0.091	0.183
Mean	31.828	**1.343**	296.894	12.216	**0.011**	0.042	0.019	0.034

datasets that feature larger numbers of class labels. Regarding the CARM DAGs, also no statistical significance in performance was detected between Two-level and All-level DAGs.

5.4.2 Multiple Path Experiments and Results

This section presents the results obtained using the DAG ensemble classification model coupled with the Multiple Path strategy. Recall that, the Multiple Path strategy was realised by utilising the probability or confidence values generated by classifier generators such as Naive Bayes and CARM, to determine whether single or multiple paths should be

followed. The results obtained when using the probability values generated during Naive Bayes classification will be considered first and then the results obtained when using the confidence values generated during CARM. Sequence of experiments was conducted included in Appendix G to identify the most appropriate threshold value σ when using Naive Bayes classification and following multiple paths in either an All-level DAG or a Two-level DAG. The outcomes from these experiments indicated that a value of $\sigma = 0.1 \times 10^{-4}$ was most appropriate in the case of an All-level DAG, while a value of $\sigma = 0.5 \times 10^{-4}$ was most appropriate in the context of the Two-level DAG.

A similar set of experiments was conducted with respect to CARM classification. These are also included in Appendix G. The outcomes from these experiments indicated that when using CARM a threshold value of $\sigma = 50$ produced the same results as when using Single Path strategy (see Table 5.2). Further experiments (not reported in this thesis) using σ values of greater than 50 were also conducted; however, this produced no change in the overall classification results compared to when using the Single Path strategy. These results indicate that the CARM classifiers forming the DAG are relatively low confidence classifiers and consequently following multiple paths may mean following very low confidence paths which in turn can be expected to result in a degradation in the overall classification accuracy, instead of improving it. Overall, from the results presented in Appendix G a value of $\sigma = 45$ was most appropriate in the case of an All-level DAG, while a value of $\sigma = 40$ was most appropriate in the context of the Two-level DAG. These results corroborate the results obtained when using *rooted* DAGs. It should also be noted here that although a best overall value for σ was identified, in the context of the Multiple Path strategy, specific best σ values can be identified for each dataset.

A comparison of the results obtained when using the DAG Multiple Path strategy with respect to both Naive Bayes classification and CARM and All-level and Two-level DAGs is presented in Table 5.5. Based on this table comparisons can be made between: (i) the usage of All-level and Two-level DAGs in the context of Naive Bayesian classification, (ii) the usage of All-level and Two-level DAGs in the context of CARM and (iii) the usage of Naive Bayes probability values and CARM confidence values for directing the Multiple Path strategy.

Starting with the Naive Bayes comparison, from the Table 5.5 it can be seen, as in the case of the Single Path strategy, that similar results are recorded regardless of whether an All-level DAG or a Two-level DAG is used. The overall performances in both cases are approximately the same in terms of average AUC. More specifically, no significant difference in performance between the two DAG models.

Regarding the CARM comparison, from Table 5.5 it can be seen, again as in the case of the Single Path strategy, that usage of the All-level DAG structure outperforms usage of the Two-level DAG structure. Recall that the number of nodes that are created for the first level in the Two-level DAG is greater than the number of nodes created for the first level in the All-level DAG. As a result the number of classifiers at the first level

TABLE 5.5: Average Accuracy and AUC values obtained using Naive Bayes and CARM classifier generation with respect to Multiple Path DAGs

Data set	Naive DAGs				CARM DAGs			
	All-level $\sigma = 0.1 \times 10^{-4}$		Two-level $\sigma = 0.5 \times 10^{-4}$		All-level $\sigma = 45$		Two-level $\sigma = 40$	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	61.50	0.32	59.14	0.33	86.81	0.43	62.44	0.45
Heart	54.88	0.35	54.19	0.35	42.51	0.18	34.71	0.15
PageBlocks	91.87	0.54	91.87	0.54	91.21	0.22	90.75	0.32
Dermatology	87.23	0.85	86.66	0.85	79.91	0.74	68.48	0.57
Glass	71.16	0.50	57.58	0.48	62.11	0.39	39.24	0.34
Zoo	93.18	0.59	93.18	0.59	88.00	0.52	88.00	0.52
Ecoli	82.26	0.38	79.93	0.37	39.51	0.28	23.22	0.13
Led	75.56	0.76	75.56	0.75	39.56	0.41	20.13	0.19
PenDigits	83.58	0.83	83.62	0.84	32.71	0.33	16.81	0.16
Mean	77.91	**0.57**	75.75	**0.57**	62.48	0.39	49.31	0.31

when using the Two-level DAG typically generate exactly the same confidence values, because of the generality of the rules, while referencing different second level nodes. Consequently when using the Two-level DAG structure there is a greater chance of following incorrect paths. However, this difference in the effectiveness between All-level and Two-level CARM DAGs, was not found to be statistically significant.

In the context of the comparison between using Naive Bayesian probability values and CARM confidence values, in the context of the Multiple Path DAG strategy the objective of the comparison was to determine the most effective classifier to be utilised with respect to Multiple Path strategy. From Table 5.5 it can clearly be observed that the usage of Naive Bayesian probability values significantly outperformed the usage of CARM confidence values when adopting the Multiple Path DAG strategy.

In addition to the classification effectiveness of the proposed Multiple Path DAG model considered above, a comparison of the efficiency of the model is presented in Table 5.6 where the results obtained for the run-time experiments with respect to Naive Bayes DAGs and CARM DAGs, coupled with the Multiple Path strategy, are reported. From the table it can be observed that the lowest classification times were obtained when using Naive Bayes and Two-level DAGs.

In conclusion, as in the case of the Single Path strategy, the above results indicate that Naive Bayes classifiers are the best choice for generating Multiple Path DAGs compared to CARM classification. With respect to the number of levels in the DAGs, there was no statistically significant difference in performance between All-level and Two-level DAGs.

TABLE 5.6: Run time results (in seconds) obtained using Naive Bayes DAGs and CARM DAGs coupled with the Multiple Path strategy

Data set	Classification Time			
	Naive		CARM	
	All-level	Two-level	All-level	Two-level
Nursery	0.625	0.626	0.601	0.611
heart	0.017	0.017	0.022	0.023
PageBlocks	0.274	0.277	0.264	0.295
Dermatology	0.024	0.021	0.023	0.025
glass	0.018	0.016	0.012	0.015
Zoo	0.010	0.008	0.015	0.015
ecoli	0.033	0.022	0.016	0.025
led	0.277	0.179	0.218	0.173
PenDigits	0.798	0.585	1.449	0.656
Mean	0.231	**0.195**	0.291	0.204

5.4.3 Comparison Between Single Path and Multiple Path Strategies

The objective of the comparison between the usage of the Single Path and Multiple Path strategies was to determine whether following more than one path within the DAG classification model could address the successive mis-classification issue noted earlier.

Starting with a comparison of the Single and Multiple Path strategies with respect to Naive Bayes DAGs Table 5.7 presents the results obtained using the All-level DAG, while Table 5.8 presents the results obtained using the Two-level DAGs. The tables were not combined because the nature of the Two-level DAGs means that datasets with much greater numbers of classes could be processed than in the case of the All-level DAGs (more specifically the Chess KRvK and Letter Recognition datasets with respect to the results presented in the tables). Note that the results presented in the tables were produced using the most appropriate σ value for all of the considered datasets (as discussed in section 5.4.2).

From Table 5.7 it can be observed that by using the Multiple Path strategy a better classification accuracy can be obtained with respect to the All-level DAG than when the Single Path strategy is adopted. More specifically, adopting the Multiple Path strategy improves the classification accuracy with respect to four of the ten datasets considered (Nursery, PageBlocks, Glass and Zoo). For one dataset (Ecoli) the Single Path strategy produced the best result. For the remaining five datasets (Heart, Dermatology, Led, PenDigits and Soybean) the same AUC results was obtained regardless of which strategy was adopted (multiple path or single path). From Table 5.8 it can also be observed that by using the Multiple Path strategy a better classification accuracy can be obtained when using Two-level DAGs with respect to some of the datasets considered in the evaluation. More specifically, adopting the Multiple Path strategy improves the classification

accuracy (over that obtained when using the Single Path strategy) with respect to two of the twelve datasets considered (Nursery and PageBlock). For another two of the data sets (Glass and Zoo) the Single Path strategy produced the best result. For the remaining eight datasets (Heart, Dermatology, Ecoli, Led, PenDigits and Soybean) the same AUC results were obtained with respect to both strategies. It is interesting to note that the effectiveness of the Multiple Path strategy was not affected by the characteristics of the considered data sets such as: number of class labels, number of examples and skewness, however, it was affected by the choice of σ value. More specifically, specific σ value can be identified for each data set to obtain a better performance. In addition, and as noted in Chapter 4, the combination technique used to distribute classes between nodes in the DAG resulted in well-defined class labels at each DAG node; consequently the number of mis-classifications is less and the effect of following multiple paths within the DAG is not highly significant. Thus, and according to the statistical tests results, no statistically significant difference in performance between Single Path and Multiple Path strategies regardless of the adopted DAG model (All-level or Two-level DAG), was noted.

TABLE 5.7: Average Accuracy and AUC values obtained using Naive Bayes All-level DAGs coupled with either the Single Path or Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$)

Data set	All-level DAG			
	Single Path		Multiple Path	
	Acc.	AUC	Acc.	AUC
Nursery	56.93	0.29	**61.50**	**0.32**
Heart	**55.22**	**0.35**	54.88	**0.35**
PageBlocks	91.83	0.53	**91.87**	**0.54**
Dermatology	**87.23**	**0.85**	**87.23**	**0.85**
Glass	69.81	0.46	**71.16**	**0.50**
Zoo	92.18	0.58	**93.18**	**0.59**
Ecoli	**84.43**	**0.41**	82.26	0.38
Led	**75.66**	**0.76**	75.56	**0.76**
PenDigits	**83.58**	**0.83**	**83.58**	**0.83**
Soybean	**90.75**	**0.92**	**90.75**	**0.92**
Mean	78.762	0.598	**79.197**	**0.604**

Regarding CARM DAGs, Table 5.9 presents the obtained results, using both the Single and Multiple Path strategies and both All-level and Two-level DAG models. From the table it can be observed that adopting the Multiple Path strategy can improve the classification accuracy with respect to some of the datasets considered. With respect to the statistical significance of these results, as in the case of the Naive Bayes DAGs, it was found that no statistically significant difference in performance between Single Path and Multiple Path strategies regardless of the adopted DAG model (All-level or Two-level DAG).

TABLE 5.8: Average Accuracy and AUC values obtained using Naive Bayes Two-level DAG coupled with either the Single Path or Multiple Path strategy ($\sigma = 0.5 \times 10^{-4}$)

Data set	Two-level DAG			
	Single Path		Multiple Path	
	Acc.	AUC	Acc.	AUC
Nursery	58.03	0.30	**59.14**	**0.33**
Heart	**54.19**	**0.35**	**54.19**	**0.35**
PageBlocks	91.83	0.53	**91.87**	**0.54**
Dermatology	**86.66**	**0.85**	**86.66**	**0.85**
Glass	**59.49**	**0.49**	57.58	0.48
Zoo	**94.18**	**0.61**	93.18	0.59
Ecoli	**80.23**	**0.37**	79.93	**0.37**
Led	**75.56**	**0.75**	**75.56**	**0.75**
PenDigits	**83.62**	**0.84**	**83.62**	**0.84**
Soybean	**90.39**	**0.92**	**90.39**	**0.92**
ChessKRvK	18.62	**0.32**	**18.79**	**0.32**
LetterRecog	**55.71**	**0.56**	55.70	**0.56**
Mean	**70.71**	0.57	70.55	**0.58**

TABLE 5.9: Average Accuracy and AUC values obtained using CARM DAGs coupled with both the Single Path and Multiple Path strategies ($\sigma = 45$ and $\sigma = 40$ were used when following multiple paths with respect to the All-level and the Two-level DAGs respectively)

Data set	All-level DAG				Two-level DAG			
	Single Path		Multiple Path		Single Path		Multiple Path	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	62.44	**0.45**
heart	53.07	**0.20**	42.51	0.18	51.70	0.20	34.71	0.15
PageBlocks	91.27	0.22	91.21	0.22	91.47	**0.45**	90.75	0.32
Dermatology	79.62	0.71	79.91	**0.74**	65.83	0.55	68.48	0.57
glass	61.71	0.36	62.11	**0.39**	59.81	0.34	39.24	0.34
Zoo	88.00	**0.52**	88.00	**0.52**	86.00	0.50	88.00	**0.52**
ecoli	32.42	0.25	39.51	**0.28**	62.94	0.23	23.22	0.13
led	42.06	**0.43**	39.56	0.41	19.28	0.18	20.13	0.19
PenDigits	41.62	**0.42**	32.71	0.33	18.95	0.19	16.81	0.16
Mean	**64.06**	**0.39**	62.48	**0.39**	60.31	0.34	49.31	0.31

Regarding run time efficiency, of course the classification time required when using the Single Path strategy is less than that required when using the Multiple Path strategy (see Tables 5.4 and 5.6).

5.4.4 Comparison Between Rooted and Non-rooted DAGs

This section presents a comparison between the operation of *rooted* and *non-rooted* DAGs in the context of multi-class classification. The objective of the comparison is to determine whether the usage of *non-rooted* DAGs could result in a better performance than the usage of *rooted* DAG as presented in the foregoing chapter (Chapter 4). Starting with comparing the effectiveness of Naive Bayes *rooted* and *non-rooted* DAGs. Table 5.10 presents the results obtained using rooted and non rooted Naive Bayes DAGs coupled with both the Single and Multiple Path strategies. From the table it can be observed that the results obtained for the various DAG models are similar (in some cases equal) for most of the datasets considered in the evaluation. More specifically, no significant difference in the effectiveness among the different DAG models regardless of the adopted classification strategy (Single or Multiple Path).

The reason behind the weakness of the *non-rooted* DAGs, is the existence of weak classifiers at the first level in the DAG that in turn affects the overall classification result. More specifically, and as noted previously, in order to classify an example using the DAG ensemble classification model all the classifiers at the first level are evaluated and a start node selected, this will be the first level node with the classifier that generates the highest probability (confidence) value with respect to the given example. It was concluded that this procedure is not sufficient to determine the best starting node among the nodes available at the first level and a technique was needed to eliminate the weak classifiers, so as to enforce the examples to be handled by only "strong" classifiers. In other words some form of breadth pruning is required (this is discussed further in the next chapter, Chapter 6).

In addition to classification effectiveness, a comparison of the efficiency of Naive Bayes *rooted* and *non-rooted* DAGs was also conducted. The outcomes are presented in Table 5.11. From the table it can be observed that the lowest generation run times were obtained using Two-level DAG, where the number of nodes to be generated are the minimum number. With respect to the Single Path classification strategy, the rooted DAG generated the lowest times, the reason for this is that both All-level and Two-level DAGs required all the classifiers at the first level to be invoked, during the classification stage, to select the best starting node among the set of nodes that exist at the first level. Regarding the Multiple Path classification strategy, the lowest run times were recorded when using the Two-level DAG.

With respect to the comparison between CARM DAGs, Table 5.12 presents the results obtained using both rooted and non rooted CARM DAGs coupled with Single and Multiple Path strategies. From the table it can be observed, that the best overall results were obtained using the rooted DAG structure. According to the conducted statistical tests, *rooted* DAG significantly outperformed Two-level DAG, while no significant difference in performance was detected between *rooted* DAG and All-level DAG. The reasons behind the weakness of the *non-rooted* DAG structures (All-level and Two-level) are: (i) the existence of weak classifiers at the first level in the DAG that affects the

TABLE 5.10: Average Accuracy and AUC values obtained using Naive Bayes *rooted* and *non-rooted* DAGs coupled with Single Path and Multiple Path strategies

Data set	Rooted DAG				All-level DAG				Two-level DAG			
	Single Path		Multiple Path $\sigma = 0.7 \times 10^{-4}$		Single Path		Multiple Path $\sigma = 0.1 \times 10^{-4}$		Single Path		Multiple Path $\sigma = 0.5 \times 10^{-4}$	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	90.26	**0.45**	90.28	**0.45**	56.93	0.29	61.50	0.32	58.03	0.30	59.14	0.33
Heart	55.91	**0.35**	55.37	**0.35**	55.22	**0.35**	54.88	**0.35**	54.19	**0.35**	54.19	**0.35**
PageBlocks	92.69	0.52	92.65	0.52	91.83	0.53	91.87	**0.54**	91.83	0.53	91.87	**0.54**
Dermatology	87.23	**0.85**	87.23	**0.85**	87.23	**0.85**	87.23	**0.85**	86.66	**0.85**	86.66	**0.85**
Glass	69.81	0.46	72.99	**0.51**	69.81	0.46	71.16	0.50	59.49	0.49	57.58	0.48
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	93.18	0.59	94.18	**0.61**	93.18	0.59
Ecoli	84.43	**0.41**	82.56	0.38	84.43	**0.41**	82.26	0.38	80.23	0.37	79.93	0.37
Led	75.66	**0.76**	75.56	**0.76**	75.66	**0.76**	75.56	**0.76**	75.56	0.75	75.56	0.75
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.62	**0.84**	83.62	**0.84**
Soybean	90.75	**0.92**	90.75	**0.92**	90.75	**0.92**	90.75	**0.92**	90.39	**0.92**	90.39	**0.92**
Mean	82.25	0.61	**82.32**	**0.62**	78.76	0.60	79.20	0.60	77.42	0.60	77.21	0.60

TABLE 5.11: Run time results (in seconds) obtained using Naive Bayes *rooted* and *non rooted* DAGs coupled with Single and Multiple Path strategies

Data set	Generation Time			Single Path Classification Time			Multiple Path Classification Time		
	Rooted	All-level	Two-level	Rooted	All-level	Two-level	Rooted	All-level	Two-level
Nursery	5.982	5.011	3.275	0.012	0.017	0.021	0.595	0.625	0.626
heart	0.333	0.261	0.262	0.001	0.001	0.001	0.015	0.017	0.017
PageBlocks	2.510	2.131	1.554	0.008	0.014	0.013	0.266	0.274	0.277
Dermatology	0.445	0.391	0.310	0.001	0.001	0.003	0.020	0.024	0.021
glass	0.539	0.490	0.290	0.001	0.001	0.003	0.016	0.018	0.016
Zoo	0.491	0.437	0.262	0.001	0.000	0.225	0.009	0.010	0.008
ecoli	1.032	0.944	0.373	0.001	0.002	0.006	0.031	0.033	0.022
led	33.701	31.803	1.095	0.011	0.013	0.034	0.261	0.277	0.179
PenDigits	264.369	244.986	4.663	0.039	0.051	0.069	0.723	0.798	0.585
soybean	1520.706	1430.897	1.505	0.009	0.012	0.037	0.068	0.078	0.060
Mean	183.011	171.735	**1.359**	**0.008**	0.011	0.041	0.200	0.215	**0.181**

TABLE 5.12: Average Accuracy and AUC values obtained using CARM *rooted* and *non-rooted* DAGs coupled with Single Path and Multiple Path strategies

Data set	Rooted DAG				All-level DAG				Two-level DAG			
	Single Path		Multiple Path $\sigma = 50$		Single Path		Multiple Path $\sigma = 45$		Single Path		Multiple Path $\sigma = 40$	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	62.44	**0.45**
Heart	53.76	**0.20**	53.76	**0.20**	53.07	**0.20**	42.51	0.18	51.70	**0.20**	34.71	0.15
PageBlocks	89.77	0.20	89.77	0.20	91.27	0.22	91.21	0.22	91.47	**0.45**	90.75	0.32
Dermatology	79.62	0.71	79.62	0.71	79.62	0.71	79.91	**0.74**	65.83	0.55	68.48	0.57
Glass	61.71	0.36	61.71	0.36	61.71	0.36	62.11	**0.39**	59.81	0.34	39.24	0.34
Zoo	88.00	**0.52**	88.00	**0.52**	88.00	**0.52**	88.00	**0.52**	86.00	0.50	88.00	**0.52**
Ecoli	32.12	0.24	32.42	0.25	32.42	0.25	39.51	**0.28**	62.94	0.23	23.22	0.13
Led	40.06	0.40	40.06	**0.43**	42.06	**0.43**	39.56	0.41	19.28	0.18	20.13	0.19
PenDigits	46.76	**0.47**	46.76	**0.47**	41.62	0.42	32.71	0.33	18.95	0.19	16.81	0.16
Mean	64.29	0.39	**64.32**	**0.40**	64.06	0.39	62.48	0.39	60.31	0.34	49.31	0.31

final classification result (the same as in the case of Naive Bayes DAGs) and (ii) the issue, discussed earlier, of several classifiers at the first level generating exactly the same confidence values, because of the generality of the rules used, but referencing different nodes at the next level. It is therefore concluded that usage of rooted DAGs provides a better framework for deciding the path (or paths) to be followed.

A comparison of the efficiency of the CARM *rooted* and *non-rooted* DAG models is presented in Table 5.13 where the results obtained for the run-time experiments with respect to CARM DAGs coupled with Single and Multiple Path strategies are reported. Again, as in the case Naive Bayes DAGs, the lowest generation times were obtained using Two-level DAG for reasons already noted. With respect to the Single Path classification strategy, the rooted DAG generated the lowest runtimes. While the lowest classification run times were recorded when using Two-level DAGs with respect to the Multiple Path strategy.

5.5 Summary

A hierarchical ensemble classification model for multi-class classification utilising a *non-rooted* Directed Acyclic Graph (DAG) structure has been presented in this chapter. The *non-rooted* DAG structure seeks to address the disadvantages of the *rooted* DAG structure (in terms of effectiveness, efficiency and scalability). An issue with respect to the *non-rooted* DAG structure is the need to determine the "starting node" (a root) from which the classification process is to commence. To this end the use of Naive Bayesian Classification or Classification Association Rule Mining (CARM) was advocated because they produce (respectively) probability and confidence values that can be utilised to determine the starting node. Two alternative classification strategies: Single Path and Multiple Path were considered. The latter was proposed to address the hierarchical drawback that if an example is mis-classified early on in the process (near the root of the hierarchy) there is no opportunity for recovery.

The operation of the *non-rooted* DAG ensemble classification model was compared with the *rooted* DAG ensemble classification model. The overall objective of the comparisons reported in this chapter was to determine whether *non-rooted* DAGs could produce a better performance than the rooted DAGs.

From the reported evaluation it was demonstrated that using Naive Bayes classifiers at the DAG nodes result in the most effective and efficient *non-rooted* DAG classification model compared to CARM *non-rooted* DAG. With respect to Naive Bayes *non-rooted* DAGs it was demonstrated that: (i) following multiple paths within the DAGs was found to be not significantly more effective than when following only a single path and (ii) reducing the number of levels in the DAG, as expected, enhanced the efficiency and hence the scalability. With respect to the usage of different numbers of levels in the *non-rooted* DAGs (Two-level or All-level), there was no statistically significant difference in the effectiveness between All-level and Two-level DAGs.

TABLE 5.13: Run time results (in seconds) obtained using CARM rooted and non rooted DAGs coupled with Single and Multiple Path strategies

Data set	Generation Time			Single Path Classification Time			Multiple Path Classification Time		
	Rooted	All-level	Two-level	Rooted	All-level	Two-level	Rooted	All-level	Two-level
Nursery	5.547	5.607	3.484	0.009	0.040	0.028	0.594	0.601	0.611
Heart	3.788	3.243	2.385	0.000	0.001	0.001	0.015	0.022	0.023
PageBlocks	3.404	3.736	2.568	0.003	0.010	0.018	0.257	0.264	0.295
Dermatology	8.542	7.010	4.048	0.001	0.002	0.004	0.023	0.023	0.025
Glass	2.281	2.650	1.341	0.000	0.001	0.004	0.015	0.012	0.015
Zoo	14.938	12.259	5.103	0.000	0.000	0.003	0.006	0.015	0.015
Ecoli	0.992	1.746	0.592	0.001	0.001	0.009	0.023	0.016	0.025
Led	25.451	42.367	1.503	0.022	0.029	0.057	0.263	0.218	0.173
PenDigits	2402.572	2593.424	88.918	0.055	0.091	0.183	1.195	1.449	0.656
Mean	274.168	296.894	**12.216**	**0.010**	0.019	0.034	0.266	0.291	**0.204**

It was discovered that an issue with the use of CARM *non-rooted* DAGs is that during the classification stage several classifiers held at the first level nodes generate exactly the same (often low) confidence value, although the nodes link to different nodes in the next level. To handle this case we tried to proceed to the next level node, which has had the highest number of links from previous level nodes (majority voting scheme). Unfortunately this did nor produce any improvement, especially when the number of nodes at the first level was large such as in the case of Two-level *non-rooted* DAGs. With respect to the Multiple Path CARM DAGs, the result was that very low confidence paths were often followed resulting in a degradation of overall classification effectiveness and there was no significant difference in performance between Single and Multiple Path strategies. With respect to the comparison between Two-level and All-level CARM DAGs, there was no statistically significant difference in the effectiveness.

Regarding the comparison between *rooted* and *non-rooted* DAGs, the results were similar and no significant difference in performance was detected between the two structures with respect to Naive Bayes DAGs. While *rooted* DAG significantly outperformed Two-level DAG and no significant difference in performance was detected between *rooted* DAG and All-level DAG with respect to CARM DAGs.

The main reason behind the weakness of the *non-rooted* DAG structures was considered to be the existence of weak classifiers at the first level in the *non-rooted* DAG that affected the overall classification results. Recall that in order to classify an example using the *non-rooted* DAG ensemble classification model, all the classifiers at the first level needed to be evaluated so that a "start node" from amongst the set of nodes available at the first level in the DAG could be identified. This node was selected according to the highest probability value for the example to be classified associated with each node. However, it was found that this procedure was not sufficient to determine the best starting node among the nodes available at the first level. A technique is needed to eliminate (prune) the weak classifiers, so that only "strong" classifiers are left, in other words breadth pruning. The concept of breadth pruning is discussed in the next chapter.

Chapter 6

The Directed Acyclic Graph (DAG) Hierarchical Classification Model with Breadth Pruning

6.1 Introduction

This chapter considers using a *non-rooted* Directed Acyclic Graph (DAG) structure to generate the desired hierarchical classification model that incorporates the concept of *breadth pruning*. As noted earlier in Chapter 4, breadth pruning can not be applied with respect to the *rooted* DAG approach. This is because the *rooted* DAG requires the inclusion of all class combinations. More specifically we cannot create a structure that commences with a root node but has nodes eliminated from the next level as this will result in "null" links from the root node. The aim of the breadth pruning is to eliminate weak classifiers that may exist at each DAG level in a *non-rooted* DAG structure, so that only strong classifiers are maintained as part of the proposed ensemble classification model. The potential advantages are: (i) improving the classification effectiveness by eliminating weak classifiers that can adversely affect classification accuracy, and (ii) reducing the complexity of the proposed model by reducing the number of nodes in the DAG model. The breadth pruning scheme was realised by utilising the AUC values generated when evaluating the internal classifiers, weak classifiers are thus identified by their low associated AUC value. Because of the effectiveness and efficiency issues associated with CARM classification, discussed in Chapters 4 and 5, only Naive Bayes classification was considered with respect to the work presented in this chapter. As in the case of the previous structures considered two alternative classification strategies were again considered: Single Path and Multiple Path. As before, the Multiple Path strategy was facilitated by the probability values generated by the Naive Bayes classifiers at the DAG nodes.

With respect to the work presented in this chapter it should be noted that two thresholds are used:

α The breadth pruning (AUC) threshold.

σ The Bayesian probability threshold for deciding whether to follow a single path or multiple paths through a generated DAG.

The rest of this chapter is organised as follow: Section 6.2 explains the generation of the DAG ensemble approach with the application of breadth pruning. While Section 6.3 considers the operation of the proposed approach. Section 6.4 presents the conducted experiments and the obtained results. Finally, a summary of the chapter is presented in Section 6.6. Note that for brevity, in the following sections the phrase DAG model is used to indicate a *non-rooted* DAG model that features breadth pruning. This should not be confused with the usage of the phrase in the foregoing two chapters.

6.2 DAG Generation with the Application of Breadth Pruning

In this section the generation of the proposed *non-rooted* DAG classification model, including the application of breadth pruning, is explained. As noted earlier, the DAG hierarchical classification model is a form of ensemble classifier. Each node in the DAG holds a classifier. Classifiers at the leaves conduct fine-grained classifications while the classifiers at non-leaf nodes conduct coarse-grained classification directed at classifying examples using groups of labels. In order to group (partition) the input data D during the hierarchy generation process, combinations techniques were used. The class groupings (sub sets) at each level are determined by finding all possible class combinations of size $|C| - i$ (where i is the level number, initially $i = 1$). As the process proceeds i is increased by one and consequently the "combination size" is decreased by one. The process continues until the combination size reaches two. The number of classifiers that need to be learned in order to generate the DAG classification model can be calculated using (6.1).

$$NumberOfClassifiers = 2^N - N - 2 \tag{6.1}$$

where N is the number of class labels in a given dataset.

Note that when N is large the number of classifiers that need to be generated will be substantial. Breadth pruning is proposed to reduce the number of classifiers, as well as a means of improving the performance of the suggested model by eliminating weak classifiers. Using the proposed breadth pruning *weak* classifiers, that may be included at each DAG level, are eliminated. The *weak* classifiers are identified by evaluating the classifiers at the first level and pruning the classifiers associated with an AUC value of less than a predefined threshold (α). The pruning process for the remaining levels involves the generation of nodes that only refer to previous level nodes (thus those associated with strong classifiers only).

Recall from Chapter 4 that the proposed DAG structure can be argued to have some similarity with the lattice structure sometimes used in frequent item set mining, however the breadth pruning threshold (α) has no relevant similarity with the downward closure property typically used in frequent item set mining.

Algorithm 14 (a and b) presents the proposed DAG generation process with breadth pruning. The input to the algorithm is the training data set D, the set of class labels C, and the breadth pruning threshold α. The DAG is created in a top down manner starting with $k = |C| - 1$ (where k is the combination size) to $k = 2$. Because of the nature of the breadth pruning mechanism the algorithm comprises two procedures: $dagFirstLevelGen$, and $dagNlevelGen$. Starting with $dagFirstLevelGen$ procedure (Algorithm 14 (a)), which is responsible for generating and pruning the first level in the DAG model. The process commences by finding the set of size k class combinations, the set C_k (line 10). We then loop through this set (line 12) and on each iteration: (i) find the set of examples D_i that feature the combination $C_i \in C_k$ (line 13); (ii) identify the training and evaluation examples, T_i and E_i (lines 14, and 15); (iii) generate a classifier G_i using T_i (line 16); (iv) evaluate the classifier G_i using E_i to produce an AUC value (line 17); (v) create a new DAG node, *node*, and add the new node to the set of accumulated level k nodes so far, $NodeSet$ (line 18). The next stage is the pruning stage, where the nodes in the $NodeSet$ are arranged according to their associated AUC values (Line 20). Then we loop through this set of ordered nodes (line 21): if the node associated with a particular AUC value is less than α (line 22), **and** its classes are included in the remaining nodes in $NodeSet$ (line 23); then the node will be pruned (line 24).

After generating and pruning the first level in the DAG model, the next step is to generate the remaining levels. The remaining levels in the DAG are created in a recursive manner using the $dagNlevelGen$ procedure (Algorithm 14 (b)). The procedure is invoked with two parameters: (i) the combination size, k, and a reference to the nodes in the current level of the DAG, $CurrentNodes$. For each call to $dagNlevelGen$ the set of size k class combinations, the set C_k, is calculated (line 30), then pruning is applied to this set (lines 31-40), where the class combination is only considered if it is a subset of one or more of the previous levels nodes' class sets. We then loop through the pruned combination set (line 42) and on each iteration: (i) find the set of training set examples T_i that feature the combination $C_i \in C_k$ (line 43), (ii) generate a classifier G_i using T_i (line 44); (iii) create a new DAG node, *node* (line 45); and (iv) add the new node to the set of accumulated level k nodes so far, $NodeSet$ (line 46). We then loop through the set of current nodes (from the previous iteration) and add a link from each current node $CurrentNode_j$ to the new node *node* whenever the set of class labels associated with the new node (C_i) is included in the set of class labels associated with a current node ($C_i \subset CurrentNodes_j.C$). Finally, if k has not yet reached 2, we repeat (line 53).

Again, because of the flexibility of: (i) the DAG structure and (ii) the combination technique used, the generation of the proposed DAG structure for any predefined number

Algorithm 14 (a) DAG Generation

```
1: INPUT: D = The input training data set, C = The set of Classes featured in D,
2:                    α = Breadth pruning threshold value
3: OUTPUT: The generated DAG
4: Start
5: k = |C| − 1
6: dagFirstLevelGeneration(k)
7: dagNlevelGen(k − 1, NodeSet)
8: End
9: procedure dagFirstLevelGen(k)
10:     C_k = Set of size k combinations in C
11:     NodeSet = {}
12:     for i = 1 to i = |C_k| do
13:         D_i = Set of examples in D that feature C_i (D_i ⊂ D)
14:         T_i= Set of examples in D_i for training G_i
15:         E_i = Set of examples in D_i for evaluating G_i
16:         G_i = Classifier for C_i built using training set T_i
17:         AUC_i = Evaluation of G_i using E_i
18:         NodeSet = NodeSet ∪ new Node(G_i, C_i, AUC_i)
19:     end for
20:     Arrange nodes in NodeSet in ascending order of associated AUC value
21:     for i = 1 to i = |NodeSet| do
22:         if (node_i.AUC < α) then
23:             if (classesCovered (node.C_i, NodeSet)) then
24:                 Delete node_i
25:             end if
26:         end if
27:     end for
28: end procedure
```

of levels can be obtained easily (using depth pruning). In this chapter we considered the generation of the all levels of the DAG model (the maximum number of levels) and generation of only two levels of the DAG (the minimum number of levels). The reason for this is that the experiments in the forgoing chapter (Chapter 5) reported that reducing the number of levels in the DAG enhanced the efficiency and scalability (in terms of the number of classes considered) of the model. For simplicity and to distinguish between these two variations and the two variations in the previous chapter, where breadth pruning was not applied, we will refer to these variations as the Max-level DAG and Min-level DAG models respectively.

Note that when using breadth pruning the eventual number of classifiers that will be generated can not be calculated in advance; however, it will clearly be less than the number of classifiers generated when breadth pruning is not used (as calculated using equation 6.1). This is evidenced by the generation and classification times reported in Section 6.4.

6.3 DAG Operation

After the model has been generated it is ready for use. As noted earlier in Chapter 5, two main challenges are associated with the operation of the proposed *non-rooted* DAG

Algorithm 14 (b) DAG Generation

```
29: procedure dagNlevelGen(k, CurrentNodes)
30:     C_k = Set of size k combinations in C
31:     for i = 1 to i = |C_k| do
32:         for j = 1 to j = |CurrentNodes| do
33:             if C_i ⊂ CurrentNodes_j.C then
34:                 flag=true, break
35:             end if
36:         end for
37:         if flag == false then
38:             Delete C_i
39:         end if
40:     end for
41:     NodeSet = {}
42:     for i = 1 to i = |C_k| do
43:         T_i = Set of training examples in D that feature C_i (T_i ⊂ D)
44:         G_i = Classifier for C_i built using training set T_i
45:         node = new Node(G_i, C_i)
46:         NodeSet = NodeSet ∪ node
47:         for j = 1 to j = |CurrentNodes| do
48:             if C_i ⊂ CurrentNodes_j.C then
49:                 CurrentNodes_j.childNodes = CurrentNodes_j.childNodes ∪ node
50:             end if
51:         end for
52:     end for
53:     if k > 2 then
54:         dagNlevelGen(k − 1, NodeSet)
55:     end if
56: end procedure
```

model: (i) how to determine the starting node among the set of nodes available at the first level, and (ii) how to address the general drawback associated with hierarchical forms of ensemble classification, the "successive mis-classification" problem. To address these issues Naive Bayesian probabilities were utilised to determine the best starting node for the DAG model, and also to decide whether a single or a multiple path should be followed at each node; consequently, two strategies are considered for classifying individual examples: Single Path and Multiple Path. The operation of Single and Multiple Path strategies are exactly the same as in the case of *non-rooted* DAG without the application of breadth pruning explained previously in Chapter 5. More specifically, the detailed procedure with respect to the Single Path classification strategy was presented in Algorithm 12 in the previous chapter (Chapter 5). While the Multiple Path classification procedure was presented in Algorithm 13 in the previous chapter (Chapter 5).

6.4 Experiments and Results

This section presents an overview of the adopted experimental set up and the evaluation results obtained. As before the effectiveness of the DAG classification model was

evaluated using twelve different data sets taken from the UCI machine learning repository [63], and pre-processed using the LUCS-KDD-DN software [23]. As in the case of the experiments presented in Chapter 5 the WaveForm and Wine datasets were not considered in the evaluation, because these datasets feature three class labels and consequently the minimum required number of DAG levels can not be generated using these datasets. Ten-fold Cross Validation (TCV) was used throughout and the evaluation measures used were average accuracy and average AUC. As before, although the results in terms of average accuracy and average AUC are both included in this section, we will discuss the results in terms of average AUC (because of the theoretical and empirical evidences that AUC is a better measure than accuracy in evaluating learning algorithms [52].

The results obtained are presented in the following sections as follows: Section 6.4.1 considers the results obtained using the Single-Path strategy and Section 6.4.2 considers the results obtained using the Multiple Path strategy. Section 6.4.3 provides a comparison between the DAG Single Path and Multiple Path strategies, while Section 6.4.4 presents a comparison between the different DAG approaches (*rooted* and *non-rooted* DAGs). Section 6.5.1 provides a comparison between the proposed DAG ensemble classification model and a number of well-established more conventional models.

6.4.1 Single Path Experiments and Results

This section presents the results obtained using the Single Path strategy with respect to the two alternative proposed DAG models, Max-level DAG and Min-level DAG. As noted in Section 6.2, a threshold α was used with respect to the breadth pruning to determine the weak classifiers that should be eliminated (if any). Experiments using a range of alternative α thresholds were conducted with respect to both the Max-level and Min-level DAGs. Details concerning these experiments are presented in Appendix H. As can be seen with reference to Appendix H the best value for α was found to be $\alpha = 0.40$. However, a specific best threshold value can be identified for each dataset. Table 6.1 presents the best results obtained for each dataset using the most appropriate α value in each case with respect to Max-level and Min-level DAGs. From the table it can be observed that the Min-level DAG generated the best overall results in terms of average AUC. More specifically, the Min-level DAG produced the best classification accuracy with respect to seven of the ten datasets considered (Nursery, Heart, Glass, Zoo, Led, PenDigits, and Soybean), although for three dataset (Led, PenDigits, and Soybean) the same result was produced using the Max-level DAG. In the remaining three cases (PageBlocks, Dermatology, and Ecoli), the Max-level DAG produced the best result. Although the Min-level DAG (the minimum number of DAG levels) generated the best overall results in terms of average AUC, according to the conducted statistical evaluation, this difference in performance between Min-level and Max-level DAGs was not statistically significant.

TABLE 6.1: The best accuracy and AUC results obtained for each dataset using different α values with respect to Max-level and Min-level DAGs

Data set	Max-level DAG			Min-level DAG		
	ACC	AUC	α	ACC	AUC	α
Nursery	79.83	0.40	0.40	91.44	**0.54**	0.70
Heart	57.01	0.39	0.20	59.91	**0.40**	0.40
PageBlocks	91.83	**0.53**	0.20	92.02	0.49	0.40
Dermatology	87.23	**0.85**	0.20	86.09	0.84	0.30
Glass	69.81	0.46	0.10	57.58	**0.48**	0.30
Zoo	92.18	0.58	0.10	93.18	**0.59**	0.40
Ecoli	84.43	**0.41**	0.10	82.40	0.40	0.40
Led	75.66	**0.76**	0.40	75.75	**0.76**	0.30
PenDigits	83.59	**0.84**	0.10	83.84	**0.84**	0.45
Soybean	90.57	**0.92**	0.40	90.04	**0.92**	0.50
Mean	81.21	0.61		**81.23**	**0.63**	

6.4.2 Multiple Path Experiments and Results

This section presents the results obtained using the DAG ensemble classification model coupled with the Multiple Path strategy. According to the experiments presented in Section 6.4.1 above $\alpha = 0.40$ was found to be the most suitable for most of the considered datasets, with respect to both the Max-level DAG and the Min-level DAG. However, as noted above, a specific α threshold value can be identified for each dataset (See Table 6.1). Consequently, two categories of Multiple Path experiments were conducted: (i) following multiple paths using $\alpha = 0.40$ with respect to the breadth pruning and (ii) following multiple paths within the DAG using the best α threshold for the data set in question (see Section 6.4.1). Details of these experiments are presented in Appendix H. From these experiments it was found that $\sigma = 0.1 \times 10^{-4}$ and $\sigma = 0.1 \times 10^{-6}$ produced the best performance with respect to Max-level and Min-level DAGs respectively, when using $\alpha = 40$. While when the most appropriate value for α was used, $\sigma = 0.1 \times 10^{-4}$ produced the best performance for both Max-level and Min-level DAGs.

Table 6.2 present a comparison between following multiple paths within the Min-level and Max-level DAGs, using a fixed breadth pruning α threshold value ($\alpha = 0.40$) and using the most appropriate α value with respect to each of the considered datasets. From the table it can be noted that, following multiple paths based on the best α value for each dataset produced a better classification performance than when using a generic α value regardless of the DAG model used. However, these differences were not found to be statistically significant according to the conducted statistical tests.

It is also note worthy that the results obtained using both the Max-level and Min-level DAGs are very similar. More specifically, the Max-level DAG produced the best AUC results for three of the considered datasets (PageBlocks, Dermatology, Glass), whilst the Min-level DAG produced the best AUC results for another three datasets

(Nursery, Heart, Ecoli). In the remaining four cases the same AUC result was produced by both DAG models. Thus, no significant difference in effectiveness between Max-level and Min-level DAGs.

TABLE 6.2: Accuracy and AUC values obtained from following multiple paths within the DAGs (Min-level and Max-level) using a fixed α value ($\alpha = 0.40$) and the best α value with respect to each dataset

Data set	α=0.40				Best α			
	Max-level		Min-level		Max-level		Min-level	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	82.32	0.41	66.28	0.39	82.32	0.41	90.02	**0.58**
Heart	56.32	0.36	59.29	**0.40**	56.25	0.37	59.64	**0.40**
PageBlocks	92.65	0.52	92.07	0.48	91.87	**0.54**	92.05	0.47
Dermatology	87.23	0.85	86.37	0.85	87.23	**0.85**	85.51	0.84
Glass	68.38	**0.50**	56.23	0.49	71.16	**0.50**	57.18	0.49
Zoo	92.18	0.58	93.18	**0.59**	92.18	**0.59**	93.18	**0.59**
Ecoli	82.26	**0.39**	80.89	**0.39**	82.56	0.38	80.89	**0.39**
Led	75.53	**0.76**	75.56	**0.76**	75.53	**0.76**	75.66	**0.76**
PenDigits	83.02	0.83	83.68	**0.84**	83.59	**0.84**	83.84	**0.84**
Soybean	90.57	**0.92**	89.86	**0.92**	90.57	**0.92**	90.04	**0.92**
Mean	81.05	0.61	78.34	0.61	81.30	0.62	80.80	**0.63**

From the above, no significant difference between identifying a different α threshold value for each dataset (the most suitable value for the dataset), and using a single generic breadth pruning α threshold value for all datasets when following multiple paths within the DAG. However, identifying a different α threshold value for each dataset was adopted because this tended to produce a better classification effectiveness.

6.4.3 Comparison Between Single Path and Multiple Path Strategies

As noted earlier in this thesis, the objective of the comparison between the Single Path and Multiple Path strategies was to determine whether following more than one path within the DAG classification model could address the successive mis-classification issue noted earlier. From experiments conducted previously, and presented above, following multiple paths based on the best α threshold, was adopted for thus purpose.

Commencing with a comparison of the Single and Multiple path strategies with respect to the Max-level DAG. Although, again from the above presented experiments, as a general rule $\sigma = 0.1 \times 10^{-4}$ had been found to produce the best performance for most of the datasets considered, a specific best value for σ can also be identified for each dataset. Table 6.3 presents the average accuracy and AUC results obtained using the Multiple Path strategy, in comparison with using a Single Path strategy (best AUC values highlighted in bold). From the table it can be observed that by using the Multiple Path strategy the operation of the proposed Max-level DAG classification model is such that the classification accuracy with respect to four of the ten datasets

considered (Nursery, PageBlocks, glass, and Zoo) was improved. For two dataset (Heart, and Ecoli) the Single Path strategy produced the best AUC result. For the remaining four datasets the same AUC results were obtained regardless of whether a Single or Multiple Path strategy was adopted.

TABLE 6.3: Average Accuracy and AUC results obtained using the Max-level DAG coupled with either a Single or a Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$)

Data set	Max-level DAG			
	Single Path		Multiple Path	
	Acc.	AUC	Acc.	AUC
Nursery	79.83	0.40	82.32	**0.41**
Heart	57.01	**0.39**	56.25	0.37
PageBlocks	91.83	0.53	91.87	**0.54**
Dermatology	87.23	**0.85**	87.23	**0.85**
Glass	69.81	0.46	71.16	**0.50**
Zoo	92.18	0.58	92.18	**0.59**
Ecoli	84.43	**0.41**	82.56	0.38
Led	75.66	**0.76**	75.53	**0.76**
PenDigits	83.59	**0.84**	83.59	**0.84**
Soybean	90.57	**0.92**	90.57	**0.92**
Mean	81.21	0.61	81.30	**0.62**

With respect to the comparison of the Single and Multiple Path strategies using the Min-level DAG. Although, again from the above presented experiments, as a general rule $\sigma = 0.1 \times 10^{-4}$ had been found to produce the best performance for most of the datasets considered, a specific best value for σ can be identified for each dataset. Table 6.4 presents the average accuracy and AUC results obtained using the Multiple Path strategy, in comparison with using a Single Path strategy (best AUC values highlighted in bold). From the table it can be observed that by using the Multiple Path strategy the operation of the proposed Min-level DAG classification model is such that the classification accuracy with respect to three of the twelve datasets considered (Nursery, Glass, and Chess KRvK) is improved. For two dataset (PageBlocks, and Ecoli) the Single Path strategy produced the best AUC result. For the remaining seven datasets the same AUC results were obtained regardless of whether a Single or Multiple Path strategy was adopted.

According to the statistical tests results, no statistically significant difference in performance between Single Path and Multiple Path strategies regardless of the adopted DAG model (Max-level or Min-level DAG). The reason for this, as noted in Chapter 4 and 5, is that the combination technique used to distribute classes between nodes in the DAG resulted in well-defined class labels at each DAG node; consequently the number of mis-classifications is less and the effect of following multiple paths within the DAG is not highly significant.

TABLE 6.4: Average Accuracy and AUC results obtained using Min-level DAG coupled with either a Single or a Multiple Path strategy ($\sigma = 0.1 \times 10^{-4}$)

Data set	Min-level DAG			
	Single Path		Multiple Path	
	Acc.	AUC	Acc.	AUC
Nursery	91.44	0.54	90.02	**0.58**
Heart	59.91	**0.40**	59.64	**0.40**
PageBlocks	92.02	**0.49**	92.05	0.47
Dermatology	86.09	**0.84**	85.51	**0.84**
Glass	57.58	0.48	57.18	**0.49**
Zoo	93.18	**0.59**	93.18	**0.59**
Ecoli	82.40	**0.40**	80.89	0.39
Led	75.75	**0.76**	75.66	**0.76**
PenDigits	83.84	**0.84**	83.84	**0.84**
Soybean	90.04	**0.92**	90.04	**0.92**
ChessKRvK	34.58	0.33	35.36	**0.36**
LetterRecog	55.85	**0.56**	55.84	**0.56**
Mean	75.223	0.596	74.934	**0.600**

The results obtained for the run-time experiments with respect to the different DAG models are presented in Table 6.5. The table presents the generation and classification time for each DAG approach. From the table it can be observed that the Min-level DAG, where depth and breadth pruning were applied, requires the least generation time. It is also clear that the Multiple Path strategy consumes more time than the Single Path strategy for both Max-level and Min-level DAGs. With respect to the Single Path strategy it can be observed that the minimum classification time was recorded when using the Max-level DAG, compared to the Min-level DAG, although both the Max-level and Min-level DAGs required that all the classifiers at the first level are invoked during the classification stage to select the best starting node among the set of nodes that exist at the first level, the number of nodes at the first level using the Max-level DAG structure is less than that featured in the Min-level DAG structure. Regarding the Multiple Path strategy the Min-level DAG required the least run time.

6.4.4 Comparison Between Different DAG Models

This section presents a comparison between the different DAG models considered in this thesis: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG and (v) Min-level DAG. The objectives of the comparison are to: (i) determine if the breadth pruning, explained in this chapter, addressed the main issue associated with *non-rooted* DAG structure, discussed earlier in the previous chapter (Chapter 5), that the existence of weak classifiers at the first level affects classification accuracy, and (ii) determine the most effective and efficient DAG structure. Although the foregoing sections and chapters established that no significant difference between Single and Multiple Path

TABLE 6.5: Classification run time results (in seconds) obtained using the DAG model (Max-level and Min-level)

Data set	Classes	Generation Time		Single Path Strategy Classification Time		Multiple Path Strategy Classification Time	
		Max level	Min level	Max level	Min level	Max level	Min level
Nursery	5	4.380	3.142	0.007	0.010	0.601	0.625
Heart	5	0.299	0.245	0.001	0.001	0.017	0.016
PageBlocks	5	2.043	1.408	0.004	0.003	0.274	0.261
Dermatology	6	0.377	0.288	0.001	0.001	0.021	0.019
Glass	7	0.415	0.261	0.000	0.001	0.017	0.013
Zoo	7	0.360	0.221	0.000	0.001	0.009	0.007
Ecoli	8	0.801	0.342	0.000	0.001	0.034	0.019
Led	10	22.142	1.097	0.005	0.022	0.266	0.163
PenDigits	10	150.180	4.508	0.033	0.047	0.658	0.526
Soybean	15	639.260	1.467	0.003	0.025	0.063	0.057
Mean		82.026	**1.298**	**0.005**	0.011	0.196	**0.171**

strategies with respect to DAG models, Multiple Path strategy tended to produce a better classification accuracy. Consequently, the results presented in this section have all been generated using the Multiple Path strategy. Table 6.6 presents the results obtained using: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG and (v) Min-level DAG; coupled with the Multiple Path strategy. From the table it can be observed that:

1. The Max-level DAG, with the application of breadth pruning, produced the best classification accuracy with respect to three of the considered datasets, in comparison with using the All-level DAG model that featured the same number of levels but without the application of breadth pruning. For the seven remaining data sets, the same result was produced using both models.

2. Similar to (1) the Min-level DAG model, with the application of breadth pruning, produced the best classification accuracy with respect to eight of the datasets considered, in comparison with using the Two-level DAG model that featured the same number of levels but without the application of breadth pruning, although for three of the dataset considered the same result was produced using both models. For two datasets the Two-level DAG produced the best results.

3. Although from (1) and (2) it seems that the application of *Breadth Pruning*, eliminating weak classifiers from the DAG, tended to produce a better classification accuracy. The differences in effectiveness, between adopting breadth pruning and not, were not found to be statistically significant.

4. Although there is a noticeable differences in the effectiveness among the considered DAG models, according to average recorded AUC, these differences were not found to be statistically significant according to the conducted Friedman test.

In addition to the comparison of effectiveness, a comparison of the efficiency of the different DAG models was conducted. The results are presented in Table 6.7. The table presents the generation and classification time for each DAG model. From the table it can be observed that the Min-level DAG, where depth and breadth pruning were applied, requires the least generation time. With respect to the Single Path strategy it can be observed that the minimum classification time was recorded when using the Max-level DAG; the reasons for this were that: (i) compared to the *rooted* DAG structure, the *non-rooted* DAG does not include a root node that needs to be evaluated; (ii) breadth pruning was applied to the first level in the Max-level DAG thus reducing the number of nodes at the first level; and (iii) compared to the Min-level DAG, although both Min-level and Max-level DAGs are required to run all the classifiers at the first level, during the classification stage to select the best starting node among the set of nodes that exist at the first level, the number of nodes at the first level using the Max-level DAG is less than that featured in the Min-level DAG. Regarding the Multiple Path strategy the Min-level DAG required the least run time.

From the above discussion we can conclude that there was no statistically significant difference in effectiveness between the different DAG models. However, the Min-level DAG model, with depth and breadth pruning, is the most efficient in comparison with: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG and (iv) Max-level DAG. Of course, scalability is another advantage of the Min-level DAG where the DAG classification model can be generated for data sets that feature larger numbers of class labels, such as the Chess KRvK and Letter Recognition data sets.

6.5 Comparison Between DAG Based Hierarchical Classification and Binary Tree Based Hierarchical Classification

This section presents a comparison between the two main models of hierarchical classification proposed in this thesis, the Binary Tree and DAG models, with respect to both the Single and Multiple Path strategies. With respect to the Binary Tree model Naive Bayes classification and data splitting was adopted for this purpose. While regarding the DAG classification model a Min-level DAG generated using Naive Bayes classifiers was adopted. The reason behind selecting these variations was that it had been previously established that they generated the best results with respect to each structure (even if the improvement was not considered statistically significant when compared to other variations). In addition these variations were the most efficient with respect to each structure.

Table 6.8 presents the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). From the table it can be clearly observed that the DAG classification model (using either the Single or the Multiple Path strategies) outperformed the Binary Tree hierarchical classification model for most of the datasets

TABLE 6.6: Accuracy and AUC results obtained using: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG, and (v) Min-level DAG coupled with Multiple Path strategy

Data set	DAG Models									
	Rooted		All-level		Two-level		Max-level		Min-level	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	90.28	0.45	61.50	0.32	59.14	0.33	82.32	0.41	90.02	**0.58**
Heart	55.37	0.35	54.88	0.35	54.19	0.35	56.25	0.37	59.64	**0.40**
PageBlocks	92.65	0.52	91.87	**0.54**	91.87	**0.54**	91.87	**0.54**	92.05	0.47
Dermatology	87.23	**0.85**	87.23	**0.85**	86.66	**0.85**	87.23	**0.85**	85.51	0.84
Glass	72.99	**0.51**	71.16	0.50	57.58	0.48	71.16	0.50	57.18	0.49
Zoo	92.18	0.58	93.18	**0.59**	93.18	**0.59**	92.18	**0.59**	93.18	**0.59**
Ecoli	82.56	0.38	82.26	0.38	79.93	0.37	82.26	0.38	80.89	**0.39**
Led	75.56	**0.76**	75.56	**0.76**	75.56	0.75	75.53	**0.76**	75.66	**0.76**
PenDigits	83.58	0.83	83.58	0.83	83.62	**0.84**	83.59	**0.84**	83.84	**0.84**
Soybean	90.75	**0.92**	90.75	**0.92**	90.39	**0.92**	90.57	**0.92**	90.04	**0.92**
Mean	82.32	0.62	79.20	0.60	77.21	0.60	81.30	0.62	80.80	**0.63**

TABLE 6.7: Run time results (in seconds) obtained using DAGs based classification approaches: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG, and (v) Min-level DAG coupled with either Single Path or Multiple Path strategies

Data set	Generation Time					Single Path Strategy Classification Time					Multiple Path Strategy Classification Time				
	Rooted level	All level	Two level	Max level	Min level	Rooted level	All level	Two level	Max level	Min level	Rooted level	All level	Two level	Max level	Min level
Nursery	5.982	5.011	3.275	4.380	3.142	0.012	0.017	0.021	0.007	0.010	0.595	0.625	0.626	0.601	0.625
Heart	0.333	0.261	0.262	0.299	0.245	0.001	0.001	0.001	0.001	0.001	0.015	0.017	0.017	0.017	0.016
PageBlocks	2.510	2.131	1.554	2.043	1.408	0.008	0.014	0.013	0.004	0.003	0.266	0.274	0.277	0.274	0.261
Dermatology	0.445	0.391	0.310	0.377	0.288	0.001	0.001	0.003	0.001	0.001	0.020	0.024	0.021	0.021	0.019
Glass	0.539	0.490	0.290	0.415	0.261	0.001	0.001	0.003	0.000	0.001	0.016	0.018	0.016	0.017	0.013
Zoo	0.491	0.437	0.262	0.360	0.221	0.001	0.000	0.225	0.000	0.001	0.009	0.010	0.008	0.009	0.007
Ecoli	1.032	0.944	0.373	0.801	0.342	0.001	0.002	0.006	0.000	0.001	0.031	0.033	0.022	0.034	0.019
Led	33.701	31.803	1.095	22.142	1.097	0.011	0.013	0.034	0.005	0.022	0.261	0.277	0.179	0.266	0.163
PenDigits	264.369	244.986	4.663	150.180	4.508	0.039	0.051	0.069	0.033	0.047	0.723	0.798	0.585	0.658	0.526
Soybean	1520.706	1430.897	1.505	639.260	1.467	0.009	0.012	0.037	0.003	0.025	0.068	0.078	0.060	0.063	0.057
Mean	183.011	171.735	1.359	82.026	**1.298**	0.008	0.011	0.041	**0.005**	0.011	0.200	0.215	0.181	0.196	**0.171**

TABLE 6.8: Average Accuracy and AUC values obtained using the Binary Tree classification model and the Min-level DAG classification model

Data set	Single Path				Multiple Path			
	Binary Tree		DAG		Binary Tree ($\sigma = 0.1 \times 10^{-6}$)		DAG ($\sigma = 0.1 \times 10^{-4}$)	
	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Nursery	90.12	0.44	**91.44**	0.54	89.09	**0.58**	90.02	**0.58**
Heart	57.70	**0.41**	**59.91**	0.40	53.77	0.36	59.64	0.40
PageBlocks	91.96	0.34	92.02	**0.49**	91.27	0.48	**92.05**	0.47
Dermatology	79.80	0.79	86.09	**0.84**	**84.60**	**0.84**	85.51	**0.84**
Glass	**63.94**	0.43	57.58	0.48	55.28	**0.51**	57.18	0.49
Zoo	**93.18**	**0.59**	**93.18**	**0.59**	92.18	0.58	**93.18**	**0.59**
Ecoli	82.31	0.36	**82.40**	**0.40**	64.15	0.27	80.89	0.39
Led	60.16	0.60	**75.75**	**0.76**	61.13	0.61	75.66	**0.76**
PenDigits	68.56	0.68	**83.84**	**0.84**	81.18	0.81	**83.84**	**0.84**
Soybean	79.55	0.81	**90.04**	**0.92**	83.71	0.83	**90.04**	**0.92**
ChessKRVK	35.18	0.27	34.58	0.33	33.88	**0.37**	**35.36**	0.36
LetRecog	39.16	0.39	**55.85**	**0.56**	53.44	0.53	55.84	**0.56**
Mean	70.14	0.51	**75.22**	**0.60**	70.31	0.56	74.93	**0.60**

considered in the evaluation, especially datasets that featured large numbers of class labels such as: Led, Pen Digits, Soybean, and Letter Recognition.

According to the conducted statistical tests, usage of the DAG structure was found to be significantly more effective with respect to the generation of the hierarchical classification model than the Binary Tree structure, regardless of the adopted classification strategy (Single or Multiple Path). The suggested reason for this is that the DAG model provides for greater flexibility than in the case of the binary tree model, because of the overlap between class groups represented by nodes at the same level in the hierarchy. The consequence of this is that the overlap partly mitigates against the early mis-classification issue. In addition, pruning the weak classifiers from the DAG model results in a better classification accuracy than in the case of the binary tree structure where all the classifiers were used.

The results obtained with respect to an associated set of run-time experiments are presented in Table 6.9. From the table it can be observed that the lowest generation and classification times were obtained when using the binary tree model; this is to be expected as it is a much less complex structure than the DAG structure. Note that the data splitting technique was used here; using either k-means clustering or *divisive* hierarchical clustering would result in higher generation times as demonstrated by the experiments reported on in Chapter 3.

6.5.1 Comparison Between The DAG Ensemble Classification Model and Conventional models

In this section a comparison between the proposed DAG classification model with depth and breadth pruning, and conventional classification models is presented. In order to

TABLE 6.9: Run time results (in seconds) obtained using the Binary Tree hierarchical model and the Min-level DAG classification model

Data set	Generation Time		Single Path Strategy Classification Time		Multiple Path Strategy Classification Time	
	BinaryTree	DAG	BinaryTree	DAG	BinaryTree	DAG
Nursery	1.048	3.142	0.008	0.010	0.576	0.625
Heart	0.229	0.245	0.001	0.001	0.020	0.016
PageBlocks	0.741	1.408	0.005	0.003	0.266	0.261
Dermatology	0.218	0.288	0.000	0.001	0.021	0.019
Glass	0.197	0.261	0.001	0.001	0.011	0.013
Zoo	0.127	0.221	0.001	0.001	0.017	0.007
Ecoli	0.197	0.342	0.001	0.001	0.024	0.019
Led	0.530	1.097	0.003	0.022	0.153	0.163
PenDigits	1.138	4.508	0.014	0.047	0.507	0.526
Soybean	0.362	1.467	0.001	0.025	0.047	0.057
ChessKRvK	1.555	70.401	0.022	0.469	1.254	1.881
LetterRecog	1.481	76.011	0.018	3.400	0.909	4.321
Mean	**0.652**	13.283	**0.006**	0.332	**0.317**	0.659

conduct a "consistent" comparison between the proposed DAG models and existing conventional models, the comparison was conducted using the same classifier generator. Consequently, a comparison between the operation of a "stand-alone" Naive Bayes classifier, Bagging of Naive Bayes classifiers and Naive Bayes DAG classification was conducted. In addition, a "non-consistent" comparison between Naive Bayes DAG and OVO SVM was conducted. The objective of this last comparison was to compare the suggested model with one of the state of the art methods for multi-class classification. For the comparison the Min-level DAG model coupled with the Multiple Path strategy was used throughout (see previous discussion about effectiveness, efficiency and scalability of this approach).

Commencing with the "consistent" comparison conducted between "stand alone" Naive Bayes classification, Bagging of Naive Bayes classifiers and Naive Bayes DAG. The results are presented in Table 6.10. From the table it can be observed that the DAG classification model improves classification accuracy with respect to six of the twelve datasets considered (Nursery, Heart, Glass, Ecoli, Led, and Chess KRvK), although for one datasets (Led) the same result was produced when Naive Bayes classification was used in stand-alone mode and with respect to bagging. For one dataset (Glass) the same result as that obtained with respect to stand alone Naive Bayes classification was obtained. For another five datasets (PageBlocks, Dermatology, PenDigits, Soybean, and LetterRecognition) the stand-alone Naive Bayes classifier produced the best result although for three dataset (PageBlocks, PenDigits, and Soybean) the same result was produced when bagging was used. For one datasets (Zoo) bagging produced the best results.

The best average AUC value, with respect to the twelve datasets considered, was obtained when using the Min-level DAG model. More specifically, the average (mean)

AUC obtained when using the Min-level DAG model on the twelve datasets was 0.60, while that obtained using a single Naive Bayes classifier or a Bagging approach produced average AUC results of 0.59 and 0.58 respectively. It is interesting to note that the proposed Min-levels DAG model tends to improve the classification effectiveness with respect to unbalanced datasets such as: Nursery, Heart, Glass, Ecoli, and ChessKRvK. It is conjectured that the combination techniques, used to distribute class labels between nodes within the DAG, helps in the handling of unbalanced datasets. More specifically, instead of letting a single classifier handle an unbalanced dataset, the combination mechanism distributes classes between DAG nodes, some nodes will handle unbalanced subsets while other nodes will handle balanced subsets. During the classification stage only a few good quality classifiers will then be used to predict the class label for a given previously unseen example, there is thus opportunity for the classifiers used to operate using balanced subsets. Consequently it is conjectured that good results are likely to be obtained. With respect to the statistical evaluation, it was found that no statistically significant difference in effectiveness between Naive Bayes classification, Bagging of Naive Bayes classifiers and Naive Bayes DAG.

TABLE 6.10: Average Accuracy and AUC values obtained using "Stand-alone" Naive Bayes classification, "Bagging" and the proposed *Min-level* DAG classification model

Data set	Classes	Naive Bayes (Single Model)		Bagging Ensemble		Min-level DAG Model	
Nursery	5	**90.22**	0.45	89.96	0.46	90.02	**0.58**
Heart	5	54.60	0.34	51.28	0.30	**59.91**	**0.40**
PageBlocks	5	**92.69**	**0.52**	92.62	**0.52**	92.02	0.49
Dermatology	6	**86.66**	**0.85**	81.00	0.81	86.09	0.84
Glass	7	**67.83**	**0.49**	55.28	0.46	57.18	**0.49**
Zoo	7	92.27	0.59	**94.27**	**0.62**	93.18	0.59
Ecoli	8	81.70	0.38	82.56	0.39	**82.40**	**0.40**
Led	10	75.59	**0.76**	75.50	**0.76**	**75.75**	**0.76**
PenDigits	10	**84.94**	**0.85**	84.57	**0.85**	83.84	0.84
Soybean	15	**91.11**	**0.93**	86.83	0.89	90.04	0.92
ChessKRvK	18	**36.32**	0.33	35.66	0.34	35.36	**0.36**
LetterRecog	26	**57.37**	**0.57**	56.93	**0.57**	55.85	0.56
Mean		**75.94**	0.59	73.87	0.58	75.14	**0.60**

The results obtained for the run-time experiments with respect to the conventional Naive Bayes classification, Bagging ensemble and the Min-level DAG are presented in Table 6.11. From the table it can be observed that the lowest generation and classification time was recorded when using the single Naive Bayes classifier. However, although the Min-level DAG model takes longer to generate, the model needs only to be generated once after which it can be used to classify data.

With respect to the "non-consistent" comparison between the operation of the Naive Bayes DAG and OVO SVM, Table 6.12 presents the results obtained in terms of average accuracy and average AUC (best results highlighted in bold font). Again, recall that

TABLE 6.11: Run time results (in seconds) obtained using "stand-alone" Naive Bayes classification, Bagging and the proposed *Min-level* DAG classification model

Data set	Naive Bayes		Bagging Ensemble		Min-level DAG	
	Gen.	Class.	Gen.	Class.	Gen.	Class.
Nursery	0.974	0.003	1.180	0.011	3.142	0.625
Heart	0.202	0.000	0.216	0.001	0.245	0.016
PageBlocks	0.676	0.001	0.775	0.005	1.408	0.261
Dermatology	0.242	0.000	0.296	0.000	0.288	0.019
Glass	0.178	0.000	0.182	0.000	0.261	0.013
Zoo	0.163	0.000	0.136	0.001	0.221	0.007
Ecoli	0.206	0.000	0.208	0.000	0.342	0.019
Led	0.529	0.002	0.547	0.004	1.097	0.163
PenDigits	1.100	0.006	1.121	0.010	4.508	0.526
Soybean	0.353	0.001	0.329	0.003	1.467	0.057
chessKRvK	1.470	0.006	1.674	0.008	70.401	1.881
LetterRecog	1.398	0.007	1.580	0.011	76.011	4.321
Mean	**0.624**	**0.002**	0.687	0.005	13.283	0.659

the results presented with respect to the Naive Bayes DAG are the results obtained when using the Min-level DAG coupled with the Multiple Path strategy and breadth pruning. From the table it can be observed that the Naive Bayes DAG produced the best classification accuracy with respect to six of the twelve datasets considered (Heart, Glass, Ecoli, Zoo, Led, and Soybean), although for one dataset (Led) the same result was produced using OVO SVM. In the remaining six cases, the OVO SVM produced the best result. The best overall results, according to the average AUC values obtained, was from using OVO SVM. It is interesting to note that, in general, the proposed Naive Bayes DAG model tended to improve the classification effectiveness with respect to unbalanced and small sized data sets (hundreds of examples) such as: Heart, Glass, Ecoli, Zoo, and Soybean. While the OVO SVM model tended to improve the classification effectiveness with respect to large sized data sets (thousands of examples) such as: Nursery, Page Blocks, Pen Digits, Chess KRvK and Letter Recognition. Note here that the last two data sets (Chess KRvK and Letter Recognition) also feature very large numbers of class labels (18 and 26 class labels).

The reported results from statistical test demonstrated that there was no statistically significant difference between Naive DAG and OVO SVM.

The results obtained for the associated run-time experiments are presented in Table 6.13. Again the table presents both the generation and classification times. From the table it can be observed, that the lowest generation and classification times were recorded when using the Min-level DAG classification model (with breadth pruning).

6.6 Summary

A hierarchical ensemble classification model for multi-class classification based on a *non-rooted* Directed Acyclic Graph (DAG) structure, with the application of *breadth pruning*,

TABLE 6.12: Average Accuracy and AUC values obtained using Naive Bayes DAG (Min-level DAG) coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier

Data set	Naive Bayes DAG		OVO SVM	
	Acc.	AUC	Acc.	AUC
Nursery	90.02	0.58	99.69	**0.64**
Heart	59.91	**0.40**	53.01	0.22
PageBlocks	92.02	0.49	92.58	**0.50**
Dermatology	86.09	0.84	88.73	**0.86**
Glass	57.18	**0.49**	72.04	0.47
Zoo	93.18	**0.59**	94.00	0.58
Ecoli	82.40	**0.40**	82.95	0.36
Led	75.75	**0.76**	75.62	**0.76**
PenDigits	83.84	0.84	98.60	**0.99**
Soybean	90.04	**0.92**	92.54	0.91
ChessKRvK	35.36	0.36	86.40	**0.81**
LetterRecog	55.85	0.56	82.92	**0.83**
Mean	75.14	0.60	84.92	**0.66**

TABLE 6.13: Run time results (in seconds) obtained using Naive Bayes DAG (Min-level DAG) coupled with the Multiple Path strategy, and One-versus-One using SVM as the base classifier

Data set	Generation Time		Classification Time	
	DAG	OVO	DAG	OVO
Nursery	3.142	10.974	0.625	0.778
Heart	0.245	0.139	0.016	0.067
PageBlocks	1.408	0.657	0.261	0.135
Dermatology	0.288	0.201	0.019	0.054
Glass	0.261	0.120	0.013	0.044
Zoo	0.221	0.066	0.007	0.028
Ecoli	0.342	0.101	0.019	0.038
Led	1.097	0.582	0.163	0.182
PenDigits	4.508	4.153	0.526	0.529
Soybean	1.467	0.386	0.057	0.151
ChessKRvK	70.401	184.841	1.881	5.293
LetterRecog	76.011	45.184	4.321	3.657
Mean	**13.283**	20.617	**0.659**	0.913

has been presented in this chapter. The aim of the breadth pruning was to eliminate weak classifiers that may exist at each DAG level, so that only strong classifiers are maintained as part of the proposed ensemble classification model. The breadth pruning scheme was realised by utilising the AUC values generated when evaluating the internal classifiers, weak classifiers were then identified by their low associated AUC value (a

threshold α was used for this purpose). Two DAG variations were considered: (i) Max-level DAG (maximum number of levels), and (ii) Min-level DAG (minimum number of levels). Two alternative classification strategies: Single Path and Multiple Path were considered. Again, the latter was facilitated by the probability values generated by Naive Bayes classifiers at the DAG nodes (a threshold σ was used to decide whether to follow a single path or not).

The operation of the proposed DAG model was compared with three well-established more conventional classification models: (i) stand-alone classification, (ii) Bagging ensemble classification, and (iii) OVO classification with SVM as the base classifiers ("non-consistent" comparison). The objective of this last comparison was to compare the operation of the proposed DAG model with one of the state of the art methods for multi-class classification. In addition, a comparison between the two main models of hierarchical classification proposed in this thesis, the Binary Tree and DAG models, with respect to both the Single and Multiple Path strategies was considered.

From the reported evaluation it was demonstrated that:

1. With respect to breadth pruning, no significant difference between identifying a different α threshold value for each data set (the most suitable value for the dataset), and using a single generic breadth pruning α threshold value for all data sets when following multiple paths within the DAG. However, identifying a different α threshold value for each data set was adopted because this tended to produce a better classification effectiveness.

2. Unlike in the case of the Binary Tree hierarchical classification model, following multiple paths within the DAG classification model was found to be not significantly more effective than when following only a single path. The reason for this was argued to be that the combination techniques used to distribute classes between nodes resulted in well-defined class labels at each DAG node, unlike the clustering algorithms that were used with respect to the Binary Tree model; consequently the mis-classification was less and the effect of following multiple paths within the DAG was not highly significant.

3. Regarding the comparison between the Max-level DAG and the Min-level DAG, there was no statistically significant difference in effectiveness. However, the Min-level DAG model, with depth and breadth pruning, is more efficient than Max-level DAG.

4. *Breadth Pruning*, did not result in a significant improvement with respect to the classification effectiveness. However, it enhanced the efficiency of the proposed DAG models. The suggested reason for the weakness of the breadth pruning mechanism is that the *auc* value, produced for the first level nodes, generated by conducting training and testing only once (as opposed to k-fold cross validation). Note that the reason for conducting training and testing only once was to limit the generation time of the DAG model.

5. Although there was a differences in the effectiveness among the considered DAG models, these differences were not found to be statistically significant, according to the conducted statistical tests.

6. According to the conducted statistical tests, usage of the DAG structure was found to be more effective with respect to the generation of the hierarchical classification model than the Binary Tree structure, regardless of the adopted classification strategy (Single or Multiple Path).

7. With respect to the comparison between Stand-alone classification, Bagging and DAG classification, although the DAG classification model improved the classification effectiveness for some of the considered datasets, the obtained improvement with respect to all of the considered datasets was not found to be statistically significant according to the Friedman test conducted.

8. Regarding the comparison between the DAG classification model and OVO SVM, the reported results from statistical test demonstrated that there was no statistically significant difference between them. However, it seems that OVO SVM is more effective than the DAG classification model for some data sets that feature large numbers of class labels such as Chess KRvK and Letter Recognition. Regarding efficiency, the DAG classification model was found to be more efficient than OVO SVM with respect to the reported generation and classification run times.

9. Finally, we can conclude that the DAG classification model does not significantly outperform the conventional methods for multi-class classification. However, it has a comparable classification effectiveness with respect to these well-established conventional methods such as stand alone Naive Bayes classification, Bagging and OVO SVM.

It is interesting to note here that, with respect to the threshold values (α and σ) an embedded procedure with a grid-search that selects the best values for the thresholds can be adopted, alternatively values for α and σ can be user-specified.

The statistical tests reported on in this chapter demonstrated that there was no statistically significant difference in operation between the DAG classification model and OVO SVM (as noted above OVO SVM outperformed the DAG model with respect to the Chess KRvK and Letter Recognition data sets). It was therefore suggested that a combination of both models (OVO SVM and DAG) might be appropriate. The expectation is that such a model will improve the classification effectiveness. However, because of the processing power required, this OVO SVM based DAG model can only be realised if it is implemented using some form of distributed or parallel computing. This is explained in further detail in the next chapter.

Chapter 7

Utilising Parallel Computing to Generate the Directed Acyclic Graph Classification Model

7.1 Introduction

One of the limiting factors of the hierarchical ensemble classification methods considered in the forgoing chapters is the computing resource required to generate the hierarchies. One potential solution is to adopt some form of multi-core or parallel computing solution. It is conjectured that this solution will:

1. Improving the efficiency of the DAG generation process, thus allowing it to be applied to datasets that feature larger numbers of class labels than have been considered so far (scalability).

2. Improve the effectiveness of the DAG classification model.

It is suggested that the latter can be realised if more effective classifiers, such as SVM classifiers, are used at each DAG node. At present "stand-alone" SVM classifiers can not be used in the context of the DAG model because they are essentially binary classifiers. To address this issue OVO SVM can be used at each DAG node. Consequently, the resulting model will be a form of ensemble of ensembles which might improve the classification effectiveness [101], but will require a great deal of processing power. This idea is motivated by the experimental evidenced presented earlier in this thesis, which indicates that the effectiveness of the base classifiers significantly affects the overall effectiveness of the proposed hierarchical ensemble classifiers. Note that using OVO SVM will entail following only a single path through the hierarchy, because no probability or confidence values can be utilised to determine whether to follow single or multiple paths.

The idea of parallelising the DAG classification model, is considered in this chapter. In the earlier chapters a number of alternative DAG models were considered, the model at which the discussion presented in this chapter is directed at is the *rooted* DAG model

from Chapter 4. Recall that the *rooted* DAG classification model is founded on the idea of arranging the classifiers into a hierarchical form by utilising a *rooted* DAG structure where each node in the *rooted* DAG holds a classifier. Classifiers at leaves act as binary classifiers while the remaining classifiers (at the root and intermediate nodes) are directed at groupings of class labels. The rest of this chapter is organised as follows. Section 7.2 provides a generic overview on parallel processing and the issues to be considered. While Section 7.3 proposes three approaches for utilising parallel processing to generate and operate the suggested DAG classification model: (i) assigning each DAG node to a process, (ii) assigning each DAG level to a process, and (iii) assigning each DAG level to a group of processes. Finally, a summary of the chapter is presented in Section 7.5.

7.2 Overview of Parallel Computing

This section provides a generic overview of the parallel processing so as to provide some background. The section is organised as follows: (i) sub-Section 7.2.1 defines parallel processing, (ii) sub-Section 7.2.2 presents the most popular categorisation of parallel architectures, (iii) sub-Section 7.2.3 presents a summarisation of the available parallel programming paradigms and (iv) sub-Section 7.2.4 briefly considers the most commonly used parallel programming language.

7.2.1 Definition

Parallel processing or parallel computing refers to the usage of multiple processing elements to work together on some common computational problem [99]. The idea is to divide a complex problem into a set of sub-problems that can be solved concurrently. Each sub-problem is assigned to a processing element. Consequently the sub-problem processes can be executed simultaneously. Not all computational problems are suited to parallel processing; for parallel processing to be an option the problem under consideration should be: (i) able to be divided into independent parts that can be solved concurrently, and (ii) solved in less time using multiple processors than when using a single processor [6]. The processors are commonly hosted in the same computer, alternatively a network of several computers, *clusters* can also be used.

7.2.2 Categorisation of Parallel Architecture

From the literature a number of categorisations have been proposed to differentiate parallel architectures. The most widely referenced classification is Flynn's Taxonomy [34]. Flynn's categorisation is founded on two factors: (i) the nature of the instruction stream, and (ii) the nature of the data stream. According to Flynn, four categories of parallel architectures can be identified:

1. **Single Instruction, Single Data (SISD).** This category refers to conventional non-parallel "sequential" computers.

2. **Single Instruction, Multiple Data (SIMD).** All the processing units execute the same instruction set, but on different portions of the data simultaneously. This kind of parallel architecture is suitable for problems that feature a high degree of regularity, such as image processing [6].

3. **Multiple Instruction, Single Data (MISD).** Each processing unit performs a different instruction set, but on the same data. This kind of parallel computer is uncommon. Although some researchers claimed that this kind of parallel architecture does not exist [87]; a suggestion example of this kind of architecture is the space shuttle flight control computer [102].

4. **Multiple Instruction, Multiple Data (MIMD).** Each processing unit performs a different instruction set on a different portion of the data. This kind of parallel computer is the most widely used.

Another well-known categorisation for parallel computers is based on the memory structure [6, 87]. According to this classification three main categories can be identified:

1. **Shared Memory.** All processing units can access a shared memory location ("global memory"), and apply changes to the data, these changes will be visible to the remaining processing units.

2. **Distributed Memory.** Each processing unit has its own memory. In this context two forms of distributed memory architecture can be identified: (i) *master-slave*, in which one process, the master process, is responsible for task distribution while the remaining processes, the slave processes (workers), work in parallel to accomplish the assigned tasks: and (ii) *peer-to-peer*, in which all processes have the same capabilities.

3. **Hybrid "Distributed-Shared" Memory.** This form of parallel architecture seeks to gain the advantages of both shared and distributed memory.

7.2.3 Parallel Programming Paradigms

The first step in parallel programming is to understand the problem to be solved using a parallel approach. After the problem has been well understood the next step is the *decomposition/partitioning* step, where the problem is split into parts that can be processed concurrently. Two main methods to obtain the desired partitions have been proposed: (i) domain (data) partitioning, and (ii) functional partitioning. Another important issue to be considered is the communication requirement, namely: (i) which tasks need to communicate with which other tasks, (ii) the type of communication (Synchronous[1] versus Asynchronous[2]) and (iii) the scope of the communication (point-to-point versus

[1]Synchronous (blocking) communication "other work must wait until the communications have completed" [6].

[2]Asynchronous (non-blocking) communication "other work can be done while communication is taking place" [6].

collective[3]). In addition to the previous issues, data dependencies, data balancing and granularity (computation/communication ratio) should be taken into consideration.

Many parallel programming paradigms are available to support the realisation of parallel programming solutions. The most commonly used paradigms are summarised as follows [15]:

1. **Master/Slave (task farming).** In this paradigm the *master* process decomposes the problem into small parts and distributes them to the remaining *slave* processes (workers). Assigning tasks to the slave processes can be performed once or in acyclic manner. Additionally, the *master* process is responsible for gathering results from slave processes. Note here that after the *master* process sends the tasks to slave processes, it can temporarily go on to behave as an additional worker process. The communication always occurs between master and slaves (not between slaves).

2. **Single Program Multiple Data (SPMD).** This paradigm is also referred to as: geometric parallelism, domain decomposition and data parallelism. In this paradigm each process performs the same instruction set (code) on different portions of the data. The data can be either: (i) generated by each process, or (ii) read from disk during the initialisation stage [15]. Communication occurs between processes.

3. **Data Pipelining (data flow).** This paradigm is fundamentally based on functional decomposition concept in which the program is split into parts that can be processed simultaneously. Each part (task) is assigned to a process. Because the processes are organised in a pipeline, the communication can be viewed as a data flow between the processes.

4. **Divide and Conquer.** This paradigm involves three main elements: (i) divide, (ii) calculate and (iii) join. This paradigm can be viewed as a tree, where the root of the tree is main problem that is divided into a number of independent sub-problems that can be handled simultaneously (the first level in the tree), each of these sub-problems is split further into sub-problems (the next level in the tree), and so on. It is interesting to note here that the difference between this paradigm and the master/slave paradigm is that in the master/slave relationship only the master process is responsible for dividing and distributing the tasks, while in the divide and conquer paradigm any process can generate sub-problems, assign them to a set of processes and collect their results (dynamic operation).

5. **Speculative Parallelism.** For complex programs that feature a high degree of data dependency, it is difficult to identify how to decompose the problem into sub-problems in an effective manner. In this case one suggested solution is to try

[3]In point-to-point communication only two tasks communicating together, one as sender and the another as receiver. While in collective communication the communication occurs among a set of processes.

what is called "speculative execution of the problem". More specifically, different solutions of the same problem can be tried concurrently, and the one that generate the best performance selected (in general this will be the one that features the shortest execution time).

6. **Hybrid Models.** This paradigm, as the name suggests, combines features from different paradigms. An example of this paradigm is to use domain (data) and functional decomposition to construct tasks.

It is interesting to mention here that:

1. "There is no best model" [6], choosing a parallel programming model depends on the problem, application domain and programmer preference.

2. The above programming models can be applied on any parallel architecture [6]. However, it has been suggested by some practitioners that the availability of resources is one of the factors that affects the choice of the parallel programming paradigm [15].

7.2.4 Parallel Programming Languages

Regarding the programming languages typically used for parallel programming, there are several available options. There are some dedicated languages but these tend to be unpopular, the reason being that programmers typically resist learning a completely new language so as to achiever parallel programming [15]. The much more popular option is to use an extension to an existing programming languages. Message passing libraries are the most commonly used mechanism for parallel programs. MPI (Message Passing Interface) is arguably the most well-known and highly used message passing library [42, 91]. MPI is essentially a standard for message passing, implementations exist in a wide variety of popular programming languages such as C, C++, Fortran, and Java.

7.3 Utilising Parallel Processing to Generate and Operate the DAG Classification Model

Having established some appropriate background in the foregoing section this section discuss the parallelisation of the proposed DAG classification model. Three solutions are suggested: (i) assigning each DAG node to a process, (ii) assigning each DAG level to a process, and (iii) assigning each DAG level to a group of processes. Note that the master/slave paradigm, with message passing, was adopted. The reasons behind the selection of this model were as follows:

1. The similarity between this model and rooted DAG topology.

2. The parallelism is very clear and explicit using the master/slave paradigm and the message passing model.

3. The combination calculation can be simply performed by the master process and then the tasks can be assigned to all other processes.

4. The necessity of a single master process to coordinate the classification stage.

5. The existence of a master process which coordinates the overall process is desirable; no processes will be lost as in the case of SPMD for example.

Each of the parallelisations is considered in the following three sub-sections. In each case the weakness associated with the parallelisation is discussed.

7.3.1 Assigning Each DAG Node to a Process

In the first suggested approach to parallelising the DAG classification problem each DAG node is assigned to a process. All class label combinations and all data portions are calculated by the master process and then distributed to the worker processes. Each worker process is responsible for creating a DAG node and generating its corresponding classifier. The master process is also responsible for the coordination of the classification process.

Algorithm 15 summarises the generation process. The input to the algorithm is a training data set D and a set of class labels C. The algorithm is divided into two main parts: a master process part and a worker process part. The master process is responsible for calculating the class combinations and data portions and sending these portions to each worker (line 12-18). The master process is also responsible for creating the root node and generating the corresponding classifier for this node (line 11). Although we present the preparation of the classes and the data in a sequential way in the algorithm, the combinations can be prepared and then distributed to all the workers simultaneously. Each worker process: (i) receives its classes and data portion from the master process and (ii) creates a DAG node and generates the corresponding classifier.

Algorithm 16 presents the classification process in detail. The input to the algorithm is a new unseen example, e, to be classified. As noted earlier the master process is responsible for the coordination of the overall classification process. More specifically, the classification process commences in the master process where the root DAG node exists. The root classifier classifies the example e and then broadcasts the initial classification result (group class) and the example e to all the worker processes once (line 10 and 11). Then only one of the worker processes will respond to the master and use its node classifier to classify the example and then send the result to the master (line 19 and 20). The worker process that will respond to the master is the one that feature the same classes as the classes sent by the root node (line 18). This process is repeated until the last level in the DAG is arrived at (this on $|C| - 2$ occasions; the number of levels in the DAG excluding the root level) where a binary classifier exists, that can then be used to assign a single class label to the example.

The issues with this approach are: (i) it is not applicable for datasets with large numbers of class labels, because of the corresponding large number of processes that

Algorithm 15 Rooted DAG Generation Adopting Parallel Processing, First Approach

```
1: INPUT
2: D = The input training dataset
3: C = The set of Classes featured in D
4: OUTPUT
5: The generated DAG
6:
7: Find out if I am MASTER or WORKER
8: if I am MASTER then
9:     k = |C| − 1
10:    C_k = Set of size k combinations in C
11:    create root node and generate its classifier
12:    for i = k to i = 2 do
13:        C_k = Set of size k combinations in C
14:        for i = 1 to |C_k| do
15:            T_i = Set of training examples in D that feature C_i (T_i ⊂ D)
16:            send C_i (portion of classes), and T_i (portion of dataset) to a WORKER
17:        end for
18:    end for
19: else (I am WORKER)
20:    receive from MASTER my classes C_i
21:    receive from MASTER my data portion T_i
22:    create node and generate classifier
23: end if
```

will be required, and (ii) during the classification stage messages are sent to all the processes, not to only a subset of processes that handle a specific level.

7.3.2 Assigning Each DAG Level to a Process

With respect to the second suggested approach, all the DAG nodes at a given level are assigned to a single process. Thus only $|C| - 1$ processes will be required. In addition, during the classification, instead of sending the results to all processes; a message will be sent to only one process each time (the process responsible for a specific level).

Algorithm 17 summarises the DAG generation process. The algorithm is similar to algorithm 15. However, instead of sending a single class combination to each process, the complete set of class combinations that represent a specific DAG level is sent to a single worker process, as well as the dataset D (line 13-15). The worker process is then responsible for: (i) receiving the classes combinations, (ii) receiving the dataset D and (iii) creating a set of DAG nodes (with the corresponding classifiers) that represent the current level (line 18-22).

Algorithm 20 presents the classification process. As before the input to the algorithm is a new unseen example, e, to be classified. Again, the master process is responsible for the coordination of the overall classification process. The classification process commences in the master process where the root DAG node exists. The root classifier, within the master process, classifies the example e. The master process will then send the initial classification result (group class), and the example e, to the worker process that handles the next level in the DAG. The levels in the DAG are assigned to an ordered

Algorithm 16 Rooted DAG Classification Adopting Parallel Processing, First Approach

INPUT
e = A new unseen example
OUTPUT
The predicted class label c for the input example e

Find out if I am MASTER or WORKER
if I am MASTER **then**
    $result$ = use my classifier to classify e
    for $i = 2$ to $NumOfLevels$ **do** (Note that Number of levels = $|C| - 1$)
        send e as collective message to all WORKERS
        send $result$ as collective message to all WORKERS
        result = receive a message from a single responded WORKER (node in our DAG)
    end for
    c = result
else (I am WORKER)
    receive from MASTER the set of classes (result)
    receive from MASTER the example e
    if my classes == result **then**
        use my node classifier to classify the example e
        send result to MASTER
    end if
end if

Algorithm 17 Rooted DAG Generation Adopting Parallel Processing, Second Approach

INPUT
D = The input training dataset
C = The set of Classes featured in D
OUTPUT
The generated DAG

Find out if I am MASTER or WORKER
if I am MASTER **then**
    $k = |C| - 1$
    C_k = Set of size k combinations in C
    create root node and generate its classifier
    for $i = k$ to $i = 2$ **do**
        C_k = Set of size k combinations in C
        send C_k a WORKER
        send the dataset to a WORKER
    end for
else (I am WORKER)
    receive from MASTER set of classes combinations classes C_k
    for $i = 1$ to $i = |C_k|$ **do**
        T_i = Set of training examples in D that feature C_i ($T_i \subset D$)
        G_i = Classifier for C_i built using training set T_i
        create node and assign the classifier
    end for
end if

set of processes (line 10 and 11). Worker process responsible for the nodes at a specific DAG level then: (i) receives the initial result (class group) from the master process and the example e (line 15 and 16), (ii) loops through its nodes (line 17) to find the node that matches the class group, (iii) uses the classifier at the identified node to classify the example and (iv) returns the result to the master process (lines 18-20). The process is repeated until the last level of the DAG is arrived at where a single class label can be assigned to the example.

Algorithm 18 Rooted DAG Classification Adopting Parallel Processing, Second Approach

```
1: INPUT
2: e = A new unseen example
3: OUTPUT
4: The predicted class label c for the input example e
5:
6: Find out if I am MASTER or WORKER
7: if I am MASTER then
8:     result = use my classifier to classify e
9:     for i = 1 to NumOfWorkers do
10:        send e, and result to WORKER associated with rank i
11:        result = receive a message from WORKER
12:    end for
13:    c = result
14: else (I am WORKER)
15:    receive from MASTER the set of classes (result)
16:    receive from MASTER e
17:    for i=1 to numOfNodes do
18:        if my classes == result then
19:            use my classifier to classify the e
20:            send result to MASTER
21:            break
22:        end if
23:    end for
24: end if
```

Note here that this approach allows considering master process during the classification stage or eliminating it so as to reduce the classification time. More specifically, at the classification stage the classification results can be passed from one process to another instead of considering a master process to coordinate this stage. Although this second approach does not require a significant number of processes, for datasets that feature a very large number of class labels the storage associated with each processor will not be sufficient to store all DAG nodes corresponding to that level when considering *cluster* parallel architecture. A potential solution is presented in the following sub-section.

7.3.3 Assigning Each DAG Level to a Group of Processes

In order to address the issues associated with the first and the second approaches, namely that dealing with large numbers of classes will still be challenging, an advanced feature of

MPI, the *groups/communicators* concept, can be utilised for thus purpose. A group is an ordered set of processes. Each process in the group is assigned a rank, ranks start at zero and continue to $N-1$, where N is the number of processes in the group. A communicator is a "handler" of a group of processors. Communicators are categorised into two kinds: (i) inter-communicators, which allow the communication between two or more groups of processors; and (ii) intra-communicators, which enable communication within a single group. It is interesting to note here that in MPI_1 only Point-to-Point communication can be established between two groups of processes using inter-communicator. While in MPI_2 the inter-communicators enable a *collective* communication within two or more groups of processes [41].

Using the MPI group management and communicator feature the idea is to assign a subset of level nodes to each process and that these processes will be kept in a single group. Figure 7.1 presents an example of the group idea. Each level in the DAG model is represented by a group of processes, each process handles a subset of level nodes. The advantages offered are: (i) the storage problem (raised with respect to the second proposed solution) will be resolved because the nodes of each DAG level will be divided among a set of processes and these processes are identified by a specific group, and (ii) during the classification stage the message will be passed only to the processes in a specific level (a processes that belongs to a specific group) not the complete set of processes.

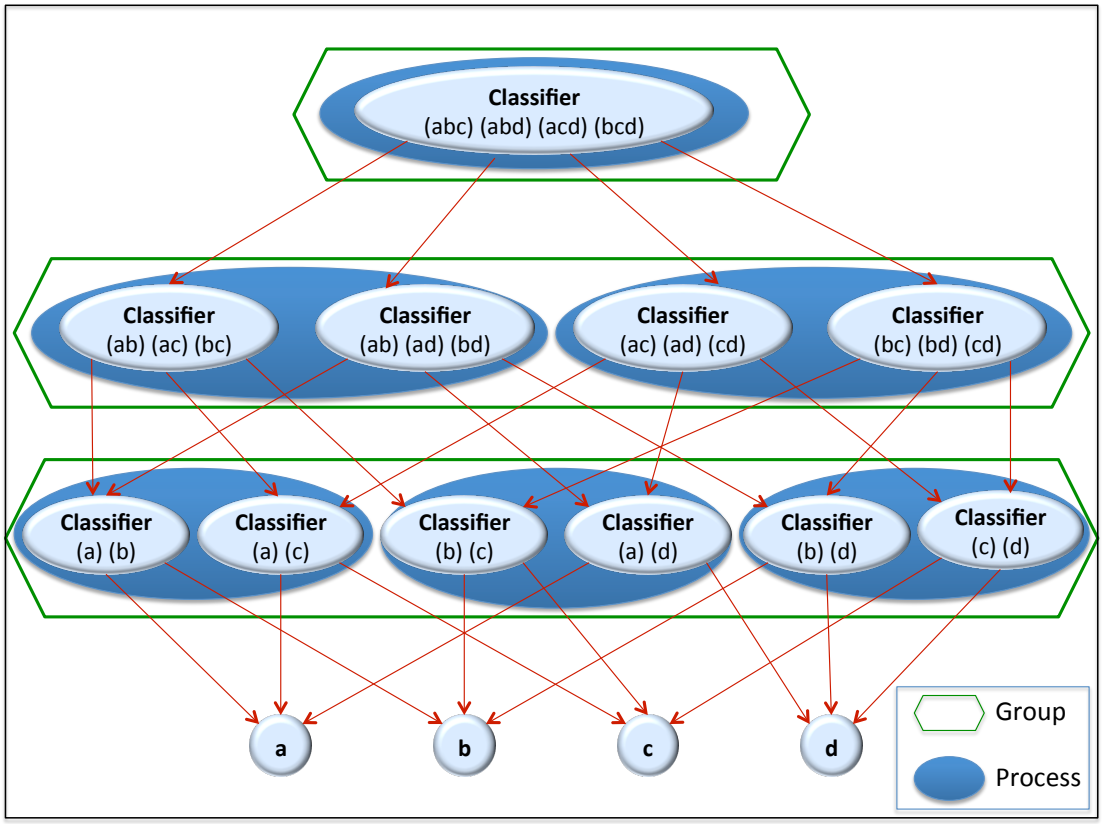


FIGURE 7.1: DAG example with parallel processing application

Algorithm 19 summarises the generation process. Again the input to the algorithm is the training data set D and the set of class labels C. The algorithm is divided into two main parts: a master process and a worker processes. The master process is responsible for: (i) determining the number of processes to be assigned to each group and (ii) identifying the combinations of class labels to be represented each level and assigning these to the process within a specific group. In more detail, the generation process starts with the creation of the first group, assigning the master process to it and creating the DAG root node with the associated classifier (lines 10 and 11). The next step is to calculate how many processes to be assigned to each level P_i (line 12-14). A group is then created for each level and assigned a specific number of processes P_i (where P_i was calculated previously (line 14)). After that the master process loops through the groups and for each group: (i) calculates the class label combinations that should exist in that group (at that level); (ii) calculates x, the number of combinations to be assigned to each process in $group_i$, by dividing the total number of combinations in that level ($|C_k|$) by the number of processes in that level (P_i); and (iii) sends x to each process in the group. Note that, although this process is described in a sequential manner, x can be sent to all processes in the group at the same time using a rooted collective operation. The second part of Algorithm 19 describes the process in a group. More specifically, a process in a group: (i) receives the class combinations and (ii) creates a set of DAG nodes with the corresponding classifiers (lines 28-32).

It is interesting to note here that the number of processes (P_i) to be assigned to each group can be calculated simply by dividing the number of available processes by the number of groups (number of levels). However, it is more efficient to assign more processes to the intermediate levels, where more DAG nodes exist.

Algorithm 20 presents the associated classification process. The input to the algorithm is a new unseen example, e, to be classified. The classification process commences with the master process; the root group where the root DAG node exists. The root classifier classifies the example e (line 8. Then we go through the remaining groups: (i) send the example e and the class group resulting from previous level (previous group) *result* to each process in the current group (line 10) and (ii) receive a result from the appropriate process in the group (line 11). The process continues until the last group (level of the DAG) is arrived at where a single class label can then be assigned to the example.

The next part of Algorithm 20 then describes the responsibility for a process in a group. A process in a group: (i) receives the initial result (class group) from the master process and the example e (line 15), (ii) loops through its nodes (line 16) to find a match for the class, (iii) use the identified classifier to classify the example (lines 17 and 18) and (iv) return the result to the master (line19).

Note here that alternatively we can send the message, the example to be classified and initial classification result (class group), from one process in a level to another

Algorithm 19 Rooted DAG Generation Adopting Parallel Processing, Third Approach

INPUT
D = The input training dataset
C = The set of Classes featured in D
OUTPUT
The generated DAG

IN THE MASTER PROCESS
$k = |C| - 1$
C_k = Set of size k combinations in C
create $group_1$ and assign the master process to it
create the root node with the associated classifier
$processes$ = find out the number of processes (number of tasks)
find out the number of combinations at each level
P_i = number of processes that will be assigned to level i (group i)
for $i = 2$ to $|C| - 1$ **do**
    create $group_i$
    assign P_i processes to $group_i$
end for
for $i = 2$ to $numOfGroups$ **do**
    C_k = Set of size k combinations in C
    $x = |C_k|$ / P_i
    for $i = 1$ to P_i **do**
        send x of C_k to a $process_i$ in the $group_i$
    end for
    $k - -$
end for
A PROCESS IN A GROUP
receive from MASTER set of classes combinations C_k
for $i = 1$ to $i = |C_k|$ **do**
    T_i = Set of training examples in D that feature C_i ($T_i \subset D$)
    G_i = Classifier for C_i built using training set T_i
    create node and assign the classifier
end for

process in the next level (point-to-point communication), the destination process can then broadcast it to all processes within the same group (same level).

7.4 Experiments and Results

In this section the results obtained from utilising parallel processing to generate and operate the rooted DAG model are presented. Note that the second approach, assigning each DAG level to a process was adopted, the reasons behind the selection of this approach were:

1. The first and the third approaches, assigning each DAG node to a process and assigning each DAG level to a group of processes, requires a larger number of processes; consequently these are more applicable when using *cluster* parallel architecture.

Algorithm 20 Rooted DAG Classification Adopting Parallel Processing, Third Approach

```
1: INPUT
2: e = A new unseen example
3: OUTPUT
4: The predicted class label c for the input example e
5:
6: IN THE MASTER PROCESS (assigned to the root group)
7: i = 1, i is the group number
8: result = use group_i process classifier to classify e
9: for i = 2 to NumOfGroups do
10:     send the example e, and result to all processes in group i
11:     result = receive result from a process
12: end for
13: c = result
14: A PROCESS IN A GROUP
15: receive e and result
16: for i=1 to numOFDAGnodes do
17:     if classes == result then
18:         result= classify e
19:         send result to root
20:         break
21:     end if
22: end for
```

2. The second approach allows for the elimination of the master process coordination during the classification stage so as to reduce the classification time.

Note here that the results presented in the earlier chapters were reported to two decimal places, however it was found that the statistical tests using two decimal places when applied with respect to the work presented in this chapter sometimes did not detect a statistical significance in performance between the different considered models whereas this was the case when results up to three significant figures were considered. Consequently the results presented in this chapter are reported to three decimal places. All experiments were conducted using a 2.7 GHz Intel Core i5 with 16 GB 1333 MHz DDR3 memory, running OS X 10.9.2 (13C64). The results in this section are organised as follows. Section 7.4.1 presents a comparison of the run time results for Naive Bayes rooted DAG with and without utilising parallel processing. While Section 7.4.2 presents the results obtained when using OVO SVM at each DAG node compared with the conventional OVO SVM.

7.4.1 Naive Byes rooted DAG Run Time Results

This section presents the results obtained from utilising parallel processing to generate the rooted DAG model. The conjectured advantage was to improve the efficiency of the rooted DAG model. Table 7.1 presents the results obtained from the run time experiments for the rooted DAG generated without using parallel processing and the rooted DAG generated using parallel processing. The table presents both generation and

classification times. With respect to the classification times using the parallel approach, two alternatives were considered: (i) the master process coordinates the classification and (ii) no master process coordinates the classification. From the table the following can be observed:

1. **Generation time.** It was noted that the generation time when utilising parallel processing, as was expected, decreased significantly for data sets that feature large number of class labels (ten classes and above) or for large sized data sets (thousands of examples). However, for small sized data sets (hundreds of examples) or that feature small number of classes (less than 10) the recorded generation times were a little bit higher than when parallel processing was not utilised. The suggested reason for this was that the initialisation of the parallel environment and the assigning of tasks to processes consumed additional time. Thus we can conclude that when we have a data set that features a small number of classes (less than 10) or a data set with hundreds of examples, as opposed to thousands, there is no advantage to be gained from using parallel processing.

2. **Classification time.** It was found that the classification time increased when using parallel processing especially when considering a master process to coordinate the classification, the reason for this is that passing messages between master process and worker processes consumes time. Classification without the master process coordination consumed less classification time. However, the classification time was still higher than that recorded when parallel processing was not utilised. The reason for this is that the process responsible for a specific level needs to loop through all its nodes to find the right node to be used to classify a given example based on the passed class subset from the previous process. In order to decrease this time, a group of processes can be used to handle a DAG level.

7.4.2 OVO SVM rooted DAG Results

This section presents the results obtained from using OVO SVM to generate the rooted DAG model. Table 7.2 presents the average accuracy and AUC results obtained using the OVO SVM rooted DAG and the conventional OVO SVM (best AUC values highlighted in bold). From the table it can be observed that the OVO SVM rooted DAG improved the classification accuracy with respect to six of the twelve considered data sets (Wine, Heart, Dermatology, Glass, Ecoli and Soybean) while the usage of the conventional OVO SVM produced the best result for only one data set (LED). In the remaining five cases both models produced the same classification accuracy. According to the Wilcoxon signed rank test, the OVO SVM rooted DAG significantly outperformed the conventional OVO SVM ($z = -2.197$, $p = 0.028 < 0.05$).

The results obtained from the run time experiments are presented in Table 7.3. From the table it can be observed that the lowest generation and classification times where recorded when using OVO SVM, the reason for this is that the OVO SVM rooted DAG

TABLE 7.1: Run time results (in seconds) obtained using parallel processing to generate the Naive Bayes rooted DAG model compared with the Naive Bayes rooted DAG model generated without using parallel processing

Data set	Generation Time		Classification Time		
	Non Parallel DAG	Parallel DAG	Non Parallel DAG	Parallel With master	Parallel Without master
WaveForm	0.20	1.00	0.00	0.05	0.03
Wine	0.19	0.27	0.00	0.00	0.00
Nursery	5.98	3.58	0.01	0.21	0.13
heart	0.33	0.46	0.00	0.01	0.01
PageBlocks	2.51	1.63	0.01	0.12	0.07
Dermatology	0.45	0.65	0.00	0.02	0.01
glass	0.54	0.81	0.00	0.02	0.01
Zoo	0.49	0.79	0.00	0.01	0.00
ecoli	1.03	1.33	0.00	0.03	0.02
lED	33.70	10.42	0.01	0.18	0.11
PenDigits	264.37	57.73	0.04	0.51	0.33
soybean	1520.71	167.69	0.01	0.31	0.22
Mean	152.54	20.53	0.01	0.12	0.08

is more complex than OVO SVM. However, the efficiency can be improved further when running the experiments on a *cluster* parallel architecture (the experiments were conducted using only four cores).

The results presented in this section evidenced that the base classifiers forming an ensemble model significantly affect the resulting ensemble accuracy.

TABLE 7.2: Average Accuracy and AUC values obtained using OVO SVM and the OVO SVM DAG

Data set	OVO SVM		OVO SVM DAG	
	Acc.	AUC	Acc.	AUC
Waveform	80.720	**0.807**	80.720	**0.807**
Wine	93.129	0.932	93.130	**0.934**
Nursery	99.691	**0.638**	99.692	**0.638**
Heart	53.008	0.219	53.106	**0.236**
PageBlocks	92.582	**0.498**	92.582	**0.498**
Dermatology	88.731	0.859	89.302	**0.863**
Glass	72.038	0.469	72.038	**0.479**
Zoo	94.000	**0.581**	94.000	**0.581**
Ecoli	82.953	0.357	82.962	**0.385**
Led	75.624	**0.757**	75.689	0.756
PenDigits	98.599	**0.986**	98.599	**0.986**
Soybean	92.543	0.912	92.722	**0.915**
Mean	85.302	0.668	85.379	**0.673**

TABLE 7.3: Run time results (in seconds) obtained using OVO SVM and the OVO SVM DAG

Data set	OVO		SVM DAG OVO	
	Generation	Classification	Generation	Classification
WaveForm	1.183	0.385	8.986	0.52
Wine	0.066	0.029	0.181	0.008
Nursery	10.974	0.778	590.569	6.149
heart	0.139	0.067	0.871	0.031
PageBlocks	0.657	0.135	95.428	1.118
Dermatology	0.201	0.054	2.636	0.049
glass	0.120	0.044	2.438	0.032
Zoo	0.066	0.028	1.218	0.021
ecoli	0.101	0.038	9.371	0.056
led	0.582	0.182	2579.976	1.22
PenDigits	4.153	0.529	63758.866	19.413
Soybean	0.386	0.151	9990.678	0.485
Mean	1.552	0.202	6420.102	2.425

7.5 Summary

This chapter has considered using parallel computing to generate the *rooted* DAG hierarchical classification model that has been presented in Chapter 4. The advantages that were expected to be obtained from adopting parallel processing for the DAG classification model are: (i) improvement of the efficiency of the Naive Bayes rooted DAG model and (ii) improvement of the effectiveness of the rooted DAG classification model by using OVO SVM at each DAG node. The *master/worker* paradigm with message passing was adopted. Three different approaches for utilising parallel processing to obtain the DAG classification model were suggested:

1. **Assigning each DAG node to a process**. In the first approach each DAG node is assigned to a process, whilst the class label combinations and data portions are calculated by the master process and then distributed to the worker processes. Each worker process is responsible for creating a DAG node and generating its corresponding classifier. The master process is also responsible for the coordination of the classification process.

2. **Assigning each DAG level to a process**. In the second approach the nodes in a single DAG level are assigned to a single process which will be responsible for an entire DAG level. During the classification, instead of sending the messages (results) to all processes intended to handle a classification problem; a message will be sent to only one process each time (thus the process associated with a specific level).

3. **Assigning each DAG level to a group of processes**. In the third approach each level in the DAG model is represented by a group of processes, each process handles a subset of the nodes at the given DAG level. During the classification stage the message (the classification result plus the example to be classified) will be passed only to the processes in a specific level (a processes that belongs to a specific group) not to all processes that handle the proposed DAG model.

Considering the: (i) number of processes that will be required and (ii) the communication between processes; the second approach was adopted in the conducted experiments. From the reported evaluation it was found that the proposed OVO SVM rooted DAG significantly outperformed the OVO SVM model which considered the state of the art method for multi-class classification. In addition using parallel processing improved the efficiency (in terms of generation time) for the Naive Bayes rooted DAG.

Chapter 8

Conclusion and Future Work

This chapter provides a summary of the research work described in this thesis, the main findings together with the contributions, and some potential directions for future work. The rest of this section is organised as follows: Section 8.1 provides a summary of the work presented while Section 8.2 presents the main findings in the context of the research question and research issues identified in Chapter 1. Finally, Section 8.3 suggests some possible directions for future work.

8.1 Summary

In this thesis a number of hierarchical ensemble classification models have been proposed as a solution to the multi-class classification problem. Such model comprises a set of base classifiers held within the nodes of the hierarchy (one classifier per node). Nodes near the root hold classifiers designed to discriminate between groups of class labels while the leaves hold classifiers designed to distinguish between individual class labels. Two types of hierarchy (structures) were considered, Binary Tree (BT) hierarchies and Directed Acyclic Graph (DAG) hierarchies.

The first structure investigated was the Binary Tree structure. In this context three different distribution techniques were considered in order to divide data between nodes during the hierarchy generation process. The distribution techniques were founded on ideas concerned with clustering and splitting techniques, namely: (i) k-means, (ii) data splitting and (iii) *divisive* hierarchical clustering. The use of two different styles of classifier at each hierarchy node was also considered: (i) single "stand-alone" classifiers and (ii) "bagging" ensemble classifiers. Three alternative classification algorithms were considered: (i) Decision tree, (ii) Naive Bayes and (iii) Classification Association Rule Mining (CARM). Two classification strategies were investigated: (i) "Single-Path" and (ii) "Multiple-Path". The second strategy was proposed to address the "successive mis-classification" issue associated with hierarchical classification, that if an example is mis-classified early on in the classification process (near the root of the hierarchy) it will continue to be mis-classified at deeper levels. In the case where more than one path is followed, a number of alternative class labels would result "candidate classes", three

alternatives for arriving at a final decision were investigated: (i) a voting mechanism; (ii) selecting the class label associated with the leaf node that features the highest probability (confidence), a process referred to as the Best Individual Probability or Confidence (BIP/BIC) process; and (iii) taking into consideration the probability (confidence) values identified along the path back to the root node to produce an *accumulated value*, a process referred to as the Normalised Accumulated Probability or Confidence (NAP/NAC) process.

The second structure investigated was the use of DAGs to generate the desired hierarchical classification model. Two alternative DAG hierarchical classification structures were proposed: (i) *rooted* DAG, and (ii) *non-rooted* DAG. To generate the DAG classification model "combination techniques" were utilised to distribute (organise) the class labels between nodes within the DAG. Again, as in the case of the Binary Tree hierarchies, three alternative classification algorithms were considered: (i) Decision tree, (ii) Naive Bayes and (iii) Classification Association Rule Mining (CARM). Two classification strategies were again considered: (i) "Single-Path" and (ii) "Multiple-Path" together with, in the later case, the three alternatives for arriving at a final classification decision as used with respect to the binary tree structure investigated: (i) Voting, (ii) BIP/BIC and (iii) NAP/NAC. To enhance the performance (effectiveness, efficiency and scalability) of the *non-rooted* DAG models two forms of pruning were considered, depth and breadth pruning. Consequently, four variations with respect to the *non-rooted DAG* were considered in this thesis: (i) All-level, (ii) Two-level, (ii) Max-level and (iv) Min-level.

In addition utilising parallel computing to generate and operate the proposed *rooted* DAG hierarchical classification model was considered. The conjecture here was that this would generate a more efficient DAG algorithm. Effectiveness may also be improved if more effective classifiers, such as SVM classifiers, are used at each DAG node. However, "stand-alone" SVM classifiers could not be used in the context of the proposed DAG models because they are essentially binary classifiers. To address this issue OVO SVM was used at each DAG node. Consequently, the resulting model could be said to be a form of ensemble of ensembles which might improve the classification effectiveness.

In the context of the evaluation, fourteen different data sets (with different numbers of class labels) were used taken from the UCI machine learning repository [63]. These were processed using the LUCS-KDD-DN data pre-processing software system [23]. Ten-fold Cross Validation (TCV) was used throughout. The evaluation measures used were average accuracy, average AUC (Area Under the receiver operating Curve) [46, 52] and run time.

The different structures, techniques, mechanisms and approaches presented in this thesis were compared with each other. The objective was to determine the best mechanisms to generate the desired hierarchical classification model. Moreover, the effectiveness of the hierarchical ensemble classification model was evaluated by comparing with more conventional existing models including:

1. A number of "stand alone" classifiers, namely: **Naive Bayes**, **Decision tree** and **CARM**. The objective was to compare the operation of the proposed model with the operation of single conventional models. Other forms of single classification model could have been selected but Naive Bayes, Decision tree and CARM were chosen because these were also used in the context of the hierarchical models considered.

2. A **Bagging** ensemble, again Naive Bayes, Decision tree, and CARM were used as the base classifiers. The objective was to compare the operation of the proposed hierarchical ensemble models with alternative forms of ensembles.

3. A **One-Versus-One** (OVO) classification mechanism using support vector machines as the base classifiers. The objectives here were: (i) to compare the proposed model with a classification mechanism founded on the use of a set of binary classifiers for solving the multi-class classification problems, and (ii) to compare the suggested model with one of the state of the art methods for multi-class classification.

To determine whether the results obtained were statistically significant a precise and comprehensive statistical analysis of the results was conducted using the Wilcoxon signed ranks test for comparing two classification models, and the Friedman test (coupled with a Nemenyi post-hoc test where appropriate) for comparing several classification models (more than two). Interesting results were obtained and these are discussed further in the next section.

8.2 Main Findings and Contributions

This section presents the main findings from the research work presented in this thesis. As initially stated in Chapter 1, the main research question to be addressed was:

"What are the most appropriate mechanisms that can be employed to generate effective hierarchical classification models?"

In order to answer this research question, the resolution of a number of subsidiary research questions were required. The work described in the thesis addresses each of these research questions as follows:

1. **Is a hierarchical classifier best arranged using a Binary Tree structure, or is it better to adopt a Directed Acyclic Graph (DAG), to effectively classify data collections that feature a large number of class labels?** According to the conducted evaluation, presented in the earlier chapters, usage of the DAG structure was found to be significantly more effective with respect to the

generation of the hierarchical classification model than the Binary Tree structure, regardless of the adopted classification strategy (Single or Multiple Path). The suggested reason for this is that the DAG model provides for greater flexibility than in the case of the binary tree model, because of the "well-defined" overlap between class groups represented by nodes at the same level in the hierarchy. The consequence of this is that the overlap partly mitigates against the early mis-classification issue. In addition, pruning the weak classifiers from the DAG model results in a better classification accuracy than in the case of the binary tree structure where all the classifiers were used.

2. **How can the nodes in a hierarchical classifier best be connected to achieve an effective classification?** With respect to the structures considered in this thesis the nodes are connected based on their class labels. More specifically, with respect to DAG structure, a node is connected to a previous node if its set of class labels is included in the set of class labels associated with that previous node. Regarding the BT structure, once one of the adopted data distribution techniques had divided the classes into two sub-groups, each sub-group was assigned to a new node. The two new nodes formed the new left and right branches emanating from the previous node. Consequently, during the classification stage a path can be followed according to the classification dictated by the internal nodes (regardless of whether a BT or DAG structure is adopted).

3. **Can a better classification accuracy be achieved by following more than one path within the hierarchy? And if so how do we decide which paths to follow?** The Multiple Path strategy was realised by utilising either Naive Bayes classification or Classification Association Rule Mining (CARM), which feature respectively probability and confidence values that can be used to determine where single or multiple paths should be followed. More specifically, more than one path was followed within the hierarchy according to a predefined threshold σ. In the case of Naive Bayes classification it was found that $0 \leq \sigma < 1$, while in the case of Classification Association Rule Mining (CARM) it was found that $0 \leq \sigma \leq 100$. Whatever the case the experimental evaluation (reported in the thesis) indicated that following multiple paths within a hierarchical classification model improved the classification effectiveness. However, the statistical evaluation of the performance indicated that:

   (a) Following multiple paths within the Binary Tree classification model was *significantly* more effective than following only a single path.

   (b) Unlike in the case of the Binary Tree hierarchical classification model, following multiple paths within the DAG classification model was not *significantly* more effective than following only a single path. The reason for this, it was argued, was that the combination techniques used to distribute classes between nodes resulted in well-defined class labels at each DAG node, unlike

the clustering algorithms that were used with respect to the Binary Tree model, consequently mis-classification was less likely and consequently the remedial strategy of following multiple paths within the DAG was not highly significant.

4. **Following on from (3) above, when adopting a multiple path strategy, how do we combine a number of possibly contradictory final classifications to provide a single end classification?** Three different mechanisms were investigated, with respect to both the BT and DAG hierarchies, for arriving at a final classification decision: (i) a voting mechanism, (ii) the BIP/BIC measure and (iii) the NAP/NAC measure. From the reported evaluation presented in Chapters 3 and 4, NAP tended to produce a better classification effectiveness. The suggested reasons are: (i) the BIP mechanism depended only on the classification result from only a single classifier (the last classifier in the path) while the NAP mechanism considered all the classifiers along the followed path, and (ii) the voting mechanism can be significantly affected by votes associated with inaccurate paths whereas the NAP mechanism assigning a specific weight to each candidate class this avoids the problem of counting votes from an inaccurate path.

5. **Following on from (3) and (4) above, will using a multiple path serve to address the "successive mis-classification" issue associated with hierarchical ensemble classification models?** The Multiple Path strategy, as already noted above, was found to be able to *partially* address the mis-classification issue. However, in order to address the successive mis-classification issue, it has been argued in this thesis that a combination of mechanisms is best adopted in addition to simply adopting a multiple path strategy. These include: (i) the use of effective classification algorithms, (ii) using a data distribution technique that can help solve the successive mis-classification problem such as the combination technique that was considered with respect to DAG hierarchies, and (iii) adopting a *pruning* procedure to eliminate the weak classifiers within the hierarchy.

6. **What is the best way of dividing up a given set of class labels between nodes (in a Binary Tree hierarchy or DAG)?** The work presented in this thesis provided a comparative study of different techniques to distribute class labels between nodes within the suggested hierarchies. Techniques founded on ideas concerned with clustering, splitting, and combinations were investigated. An issue with using clustering algorithms to distribute class labels between nodes within the hierarchy, reported in chapter 3, is that similar classes were grouped together early on in the process so that entire branches ended up dealing with very similar classes; ideally we would like individual branches to deal with very different classes so that highly discriminative classifiers can be built. Consequently, techniques based on splitting and combinations were preferred.

7. **What is the most appropriate classification algorithm to be held at individual nodes?** In addition to the efficiency and effectiveness considerations usually used to evaluate classification algorithms, a further consideration is the support that individual classification algorithms provide with respect to any multiple path strategy that might be adopted. With respect to using "stand-alone" classifier at each hierarchy node, the most effective and efficient classifier was found to be Naive Bayes classification. In addition, Naive Bayes classifiers produced probability values with which to support the proposed Multiple Path strategies. However, using ensemble of classifiers at each node was found to be more effective than using stand-alone classifiers. More specifically, using OVO SVM at each DAG node produced the most effective classification. Note here that this form of classification required utilising parallel processing, because the resulting model would be a form of ensemble of ensembles. In addition, using OVO SVM entailed following only a single path through the hierarchy.

8. **Is it indeed the case that Binary Tree hierarchical classifiers and/or DAG classifiers can be more effectively used (than when using alternative techniques) to classify data collections that feature a large number of class labels?** From the reported evaluation it was found that, when a stand-alone classifiers were utilised to generate the hierarchical classification models, no statistically significant difference was detected between the hierarchical classification models and the conventional methods for multi-class classification. While using ensemble of classifiers to generate the hierarchical classification model was found to be significantly outperforms the conventional methods for multi-class classification. More specifically, the proposed OVO SVM DAG was the most effective model for multi-class classification problems.

9. **What is the most effective classification model to be used for classifying a new data set given some characteristics regarding the new data set such as: number of examples, number of classes and skewness?**
   This question can be answered as follows:

   (a) If the data set: (i) is comprised of thousands of examples, (and/or) (ii) features large numbers of class labels (more than 10) and (iii) large numbers of processing units are available within the parallel architecture, then OVO SVM DAG is the best choice.

   (b) If the data set: (i) is comprised of thousands of examples, (and/or) (ii) features large numbers of class labels (more than 10) and (iii) a limited number of processing units is available within the parallel architecture (say four processors or cores), then the conventional OVO SVM is the best choice.

   (c) If data set: (i) is comprised of hundreds of examples and (ii) skewed, then the Naive Bayes DAG can be considered to be the best choice.

(d) If the data set features small numbers of class labels (less than 10 class labels), there was no clear insight about the best classification model. Thus, other features of the data set (such as number of examples and skewness) can be considered to identify the best classification model.

In other words, for large data sets (comprised of large number of examples or large number of classes) OVO SVM DAG and OVO SVM are recommended based on the available parallel architecture. While for small sized data sets (comprised of hundreds of examples) and skewed data sets, Naive Bayes DAG is recommended. Note that these insights were concluded based on the considered evaluation data sets.

The main contributions of the research presented in this thesis were presented in Chapter 1. These are restated here, for completeness, as follows:

1. A set of alternative techniques to distribute class labels between nodes within a Binary Tree hierarchy. With respect to the existing work on Binary Tree hierarchies, it should be noted that the most frequently used methods for dividing classes between nodes do not allow overlapping between the class groups. In this work both overlapping and non-overlapping were considered. The conjecture of allowing overlapping was that this would mitigate against the early mis-classification issue.

2. An evaluation of the use of a number of alternative classification algorithms, to generate node classifiers within a Binary Tree hierarchy. Note that existing work on Binary Tree hierarchies has mainly utilised binary classification algorithms such as SVM.

3. An "ensemble of ensembles" approach with respect to Binary Tree hierarchies. More specifically, using Bagging ensemble at each node within a binary tree hierarchy.

4. A Multiple Path strategy, which allows for more than one path to be followed within a hierarchy during the classification stage. This strategy is completely novel and it was proposed to address the "successive mis-classification" issue associated with hierarchical classification. Note here that this strategy was considered with respect to both the proposed Binary Tree and DAG hierarchies.

5. Three alternative mechanisms (Voting, BIP/BIC and NAP/NAC) for arriving at a final classification decision with respect to the Multiple Path strategy. The aim was to address the issue in the case where more than one path is followed where we end up with a number of alternative candidate class labels.

6. A unique *rooted* DAG structure for hierarchical multi-class classification.

7. A novel *non-rooted* DAG structure for hierarchical multi-class classification.

8. A novel mechanism for applying breadth pruning to the *non-rooted* DAG structure. The conjecture here was that this would improve the effectiveness and efficiency of the DAG classification model, because "weak" classifiers would be pruned.

9. A comprehensive study and statistical analysis of the proposed hierarchical ensemble classification models to identify the "best" structure, classification algorithm, data distribution technique and classification strategy to be adopted in order to obtain an effective and efficient hierarchical classification model.

10. Utilising parallel computing to generate and operate the proposed *rooted* DAG hierarchical classification model. The conjecture here was that this would generate a more efficient and effective DAG classification model that could be directed at even larger numbers of class labels.

8.3 Future Directions

The research presented in this thesis has indicated a number of promising directions for future work. Potential future directions can be itemised as follows:

1. **Utilising alternative parallel computing approaches to generate and operate the DAG classification model.** As noted in Chapter 7 parallel solution improved: (i) the efficiency of the DAG generation process, thus allowing it to be applied to data sets that feature larger numbers of class labels and (ii) the effectiveness of the DAG classification model by using OVO SVM at each DAG node. Finding and investigating more appropriate parallel approaches is thus considered to be a further fruitful avenue for future research.

2. **Investigating more effective pruning techniques with respect to DAG.** One way of applying pruning is to utilise clustering algorithms. More specifically, with respect to the DAG hierarchical ensemble classification models, clustering algorithms can be used to determine similar classes. Once the similar classes have been identified the class combinations that comprised similar classes can be pruned. Of course checking that all class labels appear in the final pruned class combinations set should be taken into account so as not to miss any class label.

3. **Using alternative data sets especially more unbalanced data sets.** As reported in Chapter 6 that DAG classification model tended to improve the classification effectiveness with respect to unbalanced datasets such as: Nursery, Heart, PageBlocks, Glass, Ecoli, and ChessKRvK. It was conjectured that the combination techniques, used to distribute class labels between nodes within the DAG, helped in the handling of unbalanced datasets. In other words, instead of letting a single classifier handle an unbalanced dataset, the combination mechanism distributes classes between DAG nodes, some nodes will handle unbalanced subsets while other nodes will handle balanced subsets. During the classification stage

only a few good quality classifiers will then be used to predict the class label for a given previously unseen example, the proposed DAG models therefore seem to operate well using balanced subsets. It is therefore suggested that this merits further investigation.

4. **Finding more effective and efficient techniques to distribute class labels between nodes within hierarchies**. Several techniques were considered in this thesis founded on ideas concerned with clustering, splitting, and combinations techniques. It was noted earlier in this thesis that the performance of the hierarchical ensemble models was significantly influenced by the adopted class partitioning technique; finding and investigating more appropriate techniques is thus considered to be a further fruitful avenue for future research.

Overall, the work presented in this thesis has produced a significant improvement with respect to a recent form of classification, namely "Hierarchical Ensemble Classification". In addition a sound foundation for future work has been provided.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499. Morgan Kaufmann, 1994.

[2] Fevzi Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. Master's thesis, Computer Engineering, Boğazici University, 1996.

[3] Mohamed Aly. Survey on multiclass classification methods. Technical report, California Institute of Technology, 2005.

[4] Maythapolnun Athimethphat and Boontarika Lerteerawong. Binary classification tree for multiclass classification with observation-based clustering. In *Proceedings of the Ninth International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON'12)*, pages 1–4. IEEE, 2012.

[5] Sami Ayramo and Tommi Karkkainen. Introduction to partitioning-based clustering methods with a robust example. Technical report, Department of Mathematical Information Technology, University of Jyväskylä, 2006.

[6] Blaise Barney. Introduction to parallel computing. URL `https://computing.llnl.gov/tutorials/parallel_comp/`. Accessed 22-March-2015.

[7] Peter Bartlett and John Shawe-taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods*, pages 43–54. MIT Press, 1999.

[8] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[9] Stephen Bay. Combining nearest neighbour classifiers through multiple feature subsets. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, pages 37–45. Morgan Kaufmann, 1998.

[10] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, Inc., 2002.

[11] Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Multiclass classification with filter trees. Preprint, available at `http://hunch.net/~jl/projects/reductions/mc_to_b/invertedTree.pdf`, June 2007.

[12] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[15] Rajkumar Buyya. *High Performance Cluster Computing: Programming and Applications*. Prentice Hall PTR, 1999.

[16] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty Third international Conference on Machine learning (ICML'06)*, pages 161–168. ACM, 2006.

[17] Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI'90)*, pages 147–149. Pitman, 1990.

[18] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011. Software available at (`http://www.csie.ntu.edu.tw/~cjlin/libsvm`).

[19] Nitesh Chawla, Steven Eschrich, and Lawrence Hall. Creating ensembles of classifiers. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'01)*, pages 580–581. IEEE, 2001.

[20] Yangchi Chen, Melba Crawford, and Joydeep Ghosh. Integrating support vector machines in a hierarchical output decomposition framework. In *Proceedings of the 2004 IEEE International Geoscience and Remote Sensing Symposium (IGARSS '04)*, pages 949–953. IEEE, 2004.

[21] Nicos Christofides. *Graph theory: an algorithmic approach*. Computer Science and Applied Mathematics. Academic press, 1975.

[22] Frans Coenen. The LUCS-KDD Apriori-T association rule mining algorithm. URL `http://www.cxc.liv.ac.uk/~frans/KDD/Software/Apriori_T/aprioriT.html`. Accessed 1-November-2015.

[23] Frans Coenen. The LUCS-KDD discretised/normalised arm and carm data library. URL `http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN`. Accessed 1-March-2013.

[24] Frans Coenen and Paul Leng. The effect of threshold values on association rule based classification accuracy. *Journal of Data and Knowledge Engineering*, 60(2):345–360, 2007.

[25] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[26] Thomas Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18:97–136, 1997.

[27] Thomas Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS'00)*, pages 1–15. Springer-Verlag, 2000.

[28] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[29] Pedro Domingos. Using partitioning to speed up specific-to-general rule induction. In *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models*, pages 29–34. AAAI Press, 1996.

[30] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[31] Margaret Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[32] Olive Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[33] Sergio Escalera, David Tax, Oriol Pujol, Petia Radeva, and Robert Duin. Subclass problem-dependent design for error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1041–1054, 2008.

[34] Michael Flynn and Kevin Rudd. Parallel architectures. *ACM Computing Surveys*, 28(1):67–70, 1996.

[35] Yoav Freund and Robert Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–80, 1999.

[36] Jerome Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[37] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[38] Manoel Fernando Alonso Gadi, Alair Pereira do Lago, and Jorn Mehnen. Data mining with skewed data. In *New Advances in Machine Learning*, pages 173–187. In Tech, 2010.

[39] Alka Gangrade and Ravindra Patel. Privacy preserving three-layer naive bayes classifier for vertically partitioned databases. *Journal of Information and Computing Science*, 8(2):119–129, 2013.

[40] Nicolas Garcia-Pedrajas and Domingo Ortiz-Boyer. Improving multiclass pattern recognition by the combination of two strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1001–1006, 2006.

[41] William Gropp, Torsten Hoefler, Rajeev Thakur, and Ewing Lusk. *Using Advanced MPI: Modern Features of the Message-Passing Interface*. MIT Press, 2014.

[42] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-passing Interface*. Scientific and Engineering Computation. MIT Press, 1999.

[43] Altay Guvenir, Gulsen Demiroz, and Nilsel Ilter. Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. *Artificial Intelligence in Medicine*, 13(3):147–165, 1998.

[44] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011.

[45] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *Special Interest Group on the Management of Data (SIGMOD) Record*, 29(2):1–12, 2000.

[46] David Hand and Robert Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

[47] Lars Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[48] Sariel Har-peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In *Proceedings of the Sixteenth Annual Conference on Neural Information Processing Systems (NIPS'03)*, pages 785–792. MIT Press, 2003.

[49] Sherif Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1994.

[50] Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *Proceeding of the Fourth International*

Conference on Intelligent Systems for Molecular Biology (ISMB'96), pages 109–115. AAAI Press, 1996.

[51] Xiaohua Hu. Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'01)*, pages 233–240. IEEE, 2001.

[52] Jin Huang and Charles Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[53] Anil Jain, M Narasimha Murty, and Patrick Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[54] Liang Kai and Zhou Ping. Using an ensemble classifier on learning evaluation for e-learning system. In *Proceedings of the 2012 International Conference on Computer Science Service System (CSSS)*, pages 538–541. IEEE, 2012.

[55] Boonserm Kijsirikul and Nitiwut Ussivakul. Multiclass support vector machines using adaptive directed acyclic graph. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02)*, pages 980–985. IEEE, 2002.

[56] Aldebaro Klautau, Nikola Jevtic, and Alon Orlitsky. On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines. *Journal of Machine Learning Research*, 4:1–15, 2003.

[57] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238. MIT Press, 1995.

[58] Shailesh Kumar, Joydeep Ghosh, and Melba Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, 5(2):210–220, 2002.

[59] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228. AAAI Press, 1992.

[60] Hansheng Lei and Venu Govindaraju. Half-against-half multi-class support vector machines. In *Proceedings of the Sixth International Workshop on Multiple Classifier Systems (MCS'05)*, pages 156–164. Springer, 2005.

[61] Thomas Leonard and John Hsu. *Bayesian Methods: An Analysis for Statisticians and Interdisciplinary Researchers*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2001.

[62] Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 369–376. IEEE, 2001.

[63] Moshe Lichman. UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`. Accessed 1-March-2013.

[64] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. An empirical comparison of decision trees and other classification methods. Technical report, Department of Statistics, University of Wisconsin, 1998.

[65] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data mining (KDD'98)*, pages 80–86. AAAI Press, 1998.

[66] Hui-Lan Luo and Zhong-Ping Liu. A review of ensemble method. In *Proceedings of the 2012 International Conference on Affective Computing and Intelligent Interaction (ICACII'12)*, pages 22–26. Information Engineering Research Institute, 2012.

[67] Kristina Machova and Peter Bednar Frantisek Barcak. A bagging method using decision trees in the role of base classifiers. *Acta Polytechnica Hungarica*, 3(2):121–132, 2006.

[68] Gjorgji Madzarov, Dejan Gjorgjevikj, and Ivan Chorbev. A multi-class svm classifier utilizing binary decision tree. *Informatica (Slovenia)*, 33(2):225–233, 2009.

[69] Oded Maimon and Lior Rokach. Ensemble of decision trees for mining manufacturing data sets. *Machine Engineering*, 4(1-2), 2004.

[70] Oded Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*. Springer, 2010.

[71] Neha Mehra and Surendra Gupta. Survey on multiclass classification methods. *International Journal of Computer Science and Information Technologies*, 4(4):572–576, 2013.

[72] Elena Montanes, Jose Barranquero, Jorge Diez, and Juan Coz. Enhancing directed binary trees for multi-class classification. *Information Sciences*, 223(0):42–55, 2013.

[73] Luis Moreira-Matias, Joao Mendes-Moreira, Joao Gama, and Pavel Brazdil. Text categorization using an ensemble classifier based on a mean co-association matrix. In *Machine Learning and Data Mining in Pattern Recognition*, pages 525–539. Springer, 2012.

[74] Sreerama Murthy. Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[75] Peter Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[76] Regina Nuzzo. Scientific method: Statistical errors. *Nature*, 506(7487):150–152, 2014.

[77] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[78] David Opitz and Jude Shavlik. Actively searching for an effective neural-network ensemble. *Connection Science*, 8(3):337–353, 1996.

[79] Nikunj Oza and Kagan Tumer. Classifier ensembles: Select real-world applications. *Information Fusion*, 9(1):4–20, 2008.

[80] John Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

[81] John Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553. MIT Press, 2000.

[82] Guangzhi Qu, Hui Zhang, and Craig Hartrick. Multi-label classification with bayes' theorem. In *Proceedings of the Fourth International Conference on Biomedical Engineering and Informatics (BMEI)*, pages 2281–2285. IEEE, 2011.

[83] John Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[84] John Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[85] John Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[86] Suju Rajan and Joydeep Ghosh. An empirical comparison of hierarchical vs. two-level approaches to multiclass problems. In *Proceedings of the Fifth International Workshop on Multiple Classifier System (MCS'04)*, pages 283–292. Springer, 2004.

[87] Thomas Rauber and Gudula Runger. *Parallel Programming: for Multicore and Cluster Systems*. Springer, 2013.

[88] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[89] Lior Rokach. Ensemble methods for classifiers. In *The Data Mining and Knowledge Discovery Handbook*, pages 957–980. Springer, 2005.

[90] Robert Schapire. Using output codes to boost multiclass learning problems. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pages 313–321. Morgan Kaufmann, 1997.

[91] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference*. Scientific and Engineering Computation. MIT Press, 1998.

[92] Patoomsiri Songsiri, Thimaporn Phetkaew, and Boonserm Kijsirikul. Enhancements of multi-class support vector machine construction from binary learners using generalization performance. Preprint, available at `http://arxiv.org/find/all/1/all:+AND+Patoomsiri+Songsiri/0/1/0/all/0/1`, September 2013.

[93] P. K. Srimani and Manjula Koti. Medical diagnosis using ensemble classifiers: A novel machine-learning approach. *Journal of Advanced Computing*, 1:9–27, 2013.

[94] David Tax and Robert Duin. Using two-class classifiers for multiclass classification. In *Proceedings of the Sixteenth International Conference on Pattern Recognition*, pages 124–127. IEEE, 2002.

[95] Krishnaiyan Thulasiraman and MNS Swamy. *Graphs: Theory and Algorithms*. John Wiley & Sons, Inc., 1992.

[96] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[97] John Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[98] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, 2000.

[99] Kumar Vipin. *Introduction to Parallel Computing*. Addison-Wesley, 2002.

[100] Volkan Vural and Jennifer Dy. A hierarchical method for multi-class support vector machines. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML'04)*, pages 105–112. ACM, 2004.

[101] Geoffrey Webb and Zijian Zheng. Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991, 2004.

[102] Wikipedia. MISD: Wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/MISD`. Accessed 20-April-2015.

[103] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[104] Jarryl Wirth and Jason Catlett. Experiments on the costs and benefits of windowing in id3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 87–99. Morgan Kaufmann, 1988.

[105] David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[106] Pengyi Yang, Yee Yang, Bing Zhou, and Albert Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):1–18, 2010.

[107] Xiaoxin Yin and Jiawei Han. CPAR: Classification based on predictive association rules. In *Proceedings of the 2003 SIAM International Conference on Data Mining (SDM'03)*, pages 331–335. SIAM, 2003.

[108] Guoqiang Zhang. Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 30(4):451–462, 2000.

[109] Zhi-Hua Zhou. Ensemble learning. In *Encyclopedia of Biometrics*, pages 270–273. Springer, 2009.

Appendix A

Evaluation data sets

In this appendix an overview of the main characteristics of the evaluation data sets is presented. Fourteen different data sets (with various numbers of class labels), taken from the University of California Irvine (UCI) machine learning repository [63], and pre-processed using the LUCS-KDD-DN software [23], were considered for evaluating the proposed hierarchical classification approaches. The pre-processing involved data discretisation and normalisation. Discretisation processes involve replacing a range of continuous values (intervals), associated with a numeric attribute, with a unique integer label (the interval label). Normalisation, in general, refers to scaling data to fall within a smaller, specified range such as 0.0-1.0 [44]. However, with respect to LUCS-KDD-DN software, normalisation refers to converting values of nominal attributes into unique integer labels [23].

The UCI machine learning repository was initiated so as to provide the machine learning community with a collection of data sets that can be used to benchmark machine learning algorithms [63]. The evaluation data sets considered in this thesis are as follows:

1. **WaveForm**. Features three types of waves, each wave is classified according two twenty-one numeric attributes. The general characteristics of the data set are provided in Table A.1.

Table A.1: Waveform Characteristics

Number of examples	5000		
Number of attributes	21		
Number of classes	3		
Number of examples per class:	Class	Num. Rec.	%
	1	1657	33.14
	2	1647	32.94
	3	1696	33.92

2. **Wine**. Derived from chemical analysis of wines and features thirteen attributes (Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Non flavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of

diluted wines, Proline), three types of wine can be identified (three classes). The general characteristics are presented in Table A.2.

Table A.2: Wine Characteristics

Number of examples	178		
Number of attributes	13		
Number of classes	3		
Number of examples per class:	Class	Num. Rec.	%
	1	59	33.15
	2	71	39.89
	3	48	26.97

3. **Nursery**. This data set was obtained from the process of ranking applications for nursery schools in Slovenia. It features eight attributes (Parents' occupation, Child's nursery (has nursery), Form of the family, Number of children, Housing conditions, Financial standing of the family, Social conditions, and Health conditions); note that some of the attribute labels are slightly eccentric and consequently difficult to interpret. Five class labels are identified: (i) Not recommend, (ii) Recommend, (iii) Very recommend, (iv) Priority and (v) Specific Priority. The general characteristics of the data set are shown in Table A.3.

Table A.3: Nursery Characteristics

Number of examples	12960		
Number of attributes	8		
Number of classes	5		
Number of examples per class:	Class	Num. Rec.	%
	1	4320	33.33
	2	2	0.02
	3	328	2.53
	4	4266	32.92
	5	4044	31.20

4. **Heart**. This data set uses 14 attributes (age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, ST depression induced by exercise relative to rest, slope of the peak exercise ST segment, number of major vessels coloured by fluoroscopy, heart rate (normal, fixed defect, reversible defect)) to predict the presence or absence of heart disease in the patient. Five different classes are identified. Four of these classes indicate the presence of heart disease, whilst the remaining class refers to the absence of heart disease. Some general characteristics for the data set are provided in Table A.4.

5. **PageBlocks**. Data set representing a primary process associated with document analysis; namely the separation of text from graphical areas. Each example represents one block of the page layout of a document obtained from a segmentation process. The data set comprises ten attributes and five classes. The attributes are

Table A.4: Heart Characteristics

Number of examples	297		
Number of attributes	13		
Number of classes	5		
Number of examples per class:	Class	Num. Rec.	%
	1	160	53.87
	2	54	18.18
	3	35	11.79
	4	35	11.79
	5	13	4.38

as follows: (i) height of the block, (ii) length of the block, (iii) area of the block (height * length), (iv) eccentricity of the block (length / height), (v) percentage of black pixels within the block, (vi) percentage of black pixels after the application of the Run Length Smoothing Algorithm (RLSA), (vii) mean number of white-black transitions, (viii) total number of black pixels in the original bitmap of the block, (ix) total number of black pixels in the bitmap of the block after the RLSA and (x) number of white-black transitions in the original bitmap of the block). The five classes are: (i) text, (ii) horizontal line, (iii) picture (iv) vertical line and (v) graphic. The general characteristics of the Page Blocks data set are given in Table A.5.

Table A.5: PageBlocks Characteristics

Number of examples	5473		
Number of attributes	10		
Number of classes	5		
Number of examples per class:	Class	Num. Rec.	%
	1	4913	89.77
	2	329	6.01
	3	28	0.51
	4	88	1.61
	5	115	2.10

6. **Dermatology**. The goal of the classification process with respect to Dermatology data set is to predict a specific type of "Erythema-Squamous" Disease. The data set comprises twelve attributes. Each example results from a patient clinical evaluation. The attributes are: (i) erythema, (ii) scaling, (iii) definite borders, (iv) itching, (v) koebner phenomenon, (vi) polygonal papules, (vii) follicular papules, (viii) oral mucosal involvement, (ix) knee and elbow involvement, (x) scalp involvement, (xi) family history and (xii) age. Six specific Erythema-Squamous types are used for the class labels: (i) Lichen Planus, (ii) Psoriasis, (iii) Seborrheic, (iv) Chronic Dermatitis, (v) Pityriasis Rosea and (vi) Pityriasis Rubra. More details concerning this data set can be found in [43]. The general characteristics of the data set are given in Table A.6.

TABLE A.6: Dermatology Characteristics

Number of examples	358		
Number of attributes	12		
Number of classes	6		
Number of examples per class:	Class	Num. Rec.	%
	1	71	19.83
	2	111	31.00
	3	60	16.76
	4	48	13.41
	5	48	13.41
	6	20	5.59

7. **Glass**. The aim of the classification process with respect to the Glass data set is to predict the type of the glass. Based on nine attributes and seven classes. The attributes are: (i) refractive index, (ii) Sodium, (iii) Magnesium, (iv) Aluminium, (v) Silicon, (vi) Potassium, (vii) Calcium, Barium and (viii) Iron. The class labels are: (i) building_windows_float_processed, (ii) building_windows_non_float_processed, (iii) vehicle_windows_float_processed, (iv) vehicle_windows_non_float_processed, (v) containers, (vi) tableware and (vii) headlamps. The general characteristics of the data are presented in Table A.7.

TABLE A.7: Glass Characteristics

Number of examples	214		
Number of attributes	10		
Number of classes	7		
Number of examples per class:	Class	Num. Rec.	%
	1	70	32.71
	2	76	35.51
	3	17	7.94
	4	0	0.00
	5	13	6.07
	6	9	4.21
	7	29	13.55

8. **Zoo**. The goal of the classification process with respect to the Zoo data set is to classify animals into seven classes. The data set comprises sixteen animal features: (i) hair, (ii) feathers, (iii) eggs, (iv) milk, (v) airborne, (vi) aquatic, (vii) predator, (viii) toothed, (ix) backbone, (x) breathes, (xi) venomous, (xii) fins, (xiii) legs, (xiv) tail, (xv) domestic and (xvi) cat_size). There are seven groups of animals identified:

   (a) Aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sea lion, squirrel, vampire, vole, wallaby, and wolf.

(b) Chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, and wren.

(c) Pit viper, sea snake, slowworm, tortoise, and tuatara.

(d) Bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, and tuna.

(e) Frog, frog, newt, and toad.

(f) Flea, gnat, honeybee, housefly, ladybird, moth, termite, and wasp.

(g) Clam, crab, crayfish, lobster, octopus, scorpion, sea wasp, slug, starfish, and worm.

The general characteristics of the Zoo data set are given in Table A.8.

TABLE A.8: Zoo Characteristics

Number of examples	101		
Number of attributes	16		
Number of classes	7		
Number of examples per class:	Class	Num. Rec.	%
	1	41	40.59
	2	20	19.80
	3	5	4.95
	4	13	12.87
	5	4	3.96
	6	8	7.92
	7	10	9.90

9. **Ecoli**. The Ecoli data set comprises eight attributes and eight classes. The aim of classification when applied to this data set is to classify proteins into their various cellular localisation sites based on their amino-acid sequences [50]. The eight attributes are: (i) sequence name, (ii) mcg, (iii) lip, (iv) chg, (v) aac, (vi) alm1, (vii) alm2, and (viii) gvh. The eight localisation sites (classes) are as follows: (i) cytoplasm, (ii) inner membrane without signal sequence, (iii) perisplasm, (iv) inner membrane un-cleavable signal sequence, (v) outer membrane, (vi) outer membrane lipoprotein, (vii) inner membrane and (vii) lipoprotein inner membrane cleavable signal sequence. More details concerning this data set are available in [50]. Some general characteristics associated with the Ecoli data set are provided in Table A.9.

10. **Led**. This data set feature seven boolean attributes, because the Light-Emitting Diodes (LED) display is assumed to comprise seven diodes. The boolean value associated with each attribute indicates whether the corresponding diode is on or off. Based on the seven attributes (light1, light2, ..., light7) ten "concepts are identified (the set of decimal digits 0-9). These concepts are the ten classes featured in the data set. The general characteristics of the data set are listed in Table A.10.

TABLE A.9: Ecoli Characteristics

Number of examples	336		
Number of attributes	7		
Number of classes	8		
Number of examples per class:	Class	Num. Rec.	%
	1	143	45.56
	2	77	22.92
	3	52	15.48
	4	35	10.42
	5	20	5.95
	6	5	1.49
	7	2	0.60
	8	2	0.60

TABLE A.10: Led Characteristics

Number of examples	3200		
Number of attributes	7		
Number of classes	10		
Number of examples per class:	Class	Num. Rec.	%
	1	329	10.28
	2	350	10.94
	3	313	9.78
	4	307	9.59
	5	312	9.75
	6	313	9.78
	7	301	9.41
	8	314	9.81
	9	327	10.22
	10	334	10.44

11. **Pen-Based Recognition of Handwritten Digits Data Set (PenDigits)**. As the name indicates, the aim of the classification process with respect to this data set is to classify an example according to the set of digits 0 through to 9 (thus 10 classes). The data set is based on sixteen attributes describing x and y points collected as each digit was written. More details concerning this data set can be found in [2]. Some general characteristics concerning the data set are provided in Table A.11.

12. **Soybean**. The aim of the classification process with respect to the Soybean data set is to classify an instance into one of fifteen soybean diseases. The data set comprises thirty five attributes (date, plant-stand, precip, temp, hail, crop-hist, area-damaged, severity, seed-tmt, germination, plant-growth, leaves, leaf-spots-halo, leaf-spots-marg, leaf-spot-size, leaf-shread, leaf-malf, leaf-mild, stem, lodging, stem-cankers, canker-lesion, fruiting-bodies, external decay, mycelium,

TABLE A.11: PenDigits Characteristics

Number of examples	10992		
Number of attributes	16		
Number of classes	10		
Number of examples per class:	Class	Num. Rec.	%
	1	1143	10.40
	2	1143	10.40
	3	1144	10.41
	4	1055	9.60
	5	1144	10.41
	6	1055	9.60
	7	1056	9.61
	8	1142	10.39
	9	1055	9.60
	10	1055	9.60

int-discolor, sclerotia, fruit-pods, fruit-pods, fruit spots, seed, mold-growth, seed-discolor, shriveling and roots). The class set is: diaporthe-stem-canker, charcoal-rot, rhizoctonia-root-rot, phytophthora-rot, brown-stem-rot, powdery-mildew, downy-mildew, brown-spot, bacterial-blight, bacterial-pustule, purple-seed-stain, anthracnose, phyllosticta-leaf-spot, alternarialeaf-spot and frog-eye-leaf-spot,). Some general characteristics of the data set are provided in Table A.12.

TABLE A.12: Soybean Characteristics

Number of examples	562		
Number of attributes	35		
Number of classes	15		
Number of examples per class:	Class	Num. Rec.	%
	1	20	3.56
	2	20	3.56
	3	20	3.56
	4	20	3.56
	5	44	7.83
	6	20	3.56
	7	20	3.56
	8	92	16.37
	9	20	3.56
	10	20	3.56
	11	20	3.56
	12	44	7.83
	13	20	3.56
	14	91	16.19
	15	91	16.19

13. **Chess King-Rook v. King (Chess KRvK)**. The aim of the classification process with respect to this data set is to predict the optimal "depth of win" for the white player in 0 to 16 moves inclusive (or a draw) using only the white king and rook and the black king chess pieces. The data set comprises six attributes: (i) White King file, (ii) White King rank, (iii) White Rook file, (iv) White Rook

rank, (v) Black King file and (vi) Black King; and eighteen classes (draw, zero, one, two, three, ..., sixteen). Some general characteristics of the data set are provided in Table A.13.

TABLE A.13: Chess KRvK Characteristics

Number of examples	28056		
Number of attributes	6		
Number of classes	18		
Number of examples per class:	Class	Num. Rec.	%
	1	2796	9.97
	2	27	0.1
	3	78	0.28
	4	246	0.88
	5	81	0.29
	6	198	0.71
	7	471	1.68
	8	592	2.11
	9	683	2.43
	10	1433	5.11
	11	1712	6.1
	12	1985	7.08
	13	2854	10.17
	14	3597	12.82
	15	4194	14.95
	16	4553	16.23
	17	2166	7.72
	18	390	1.39

14. **Letter Recognition**. This data set describes black-and-white rectangular pixel displays, each representing one of the twenty six capital letters available in the English alphabet. The data set comprises sixteen attributes related to x and y coordinates (i) x-box, (ii) y-box, (iii) width, (iv) high, (v) onpix, (vi) x-bar, (vii) y-bar, (viii) x2bar, (ix) y2bar, (x) xybar, (xi) x2ybr, (xii) xy2br, (xiii) x-ege, (xiv) xegvy, (xv) y-ege and (cvi) yegvx. The class labels are the set of twenty six capital letters (A, B, C, D, ..., Z). The general characteristics of the data set are provided in Table A.14.

TABLE A.14: Letter Recognition Characteristics

Number of examples	20000		
Number of attributes	16		
Number of classes	26		
Number of examples per class:	Class	Num. Rec.	%
	1	789	3.95
	2	766	3.83
	3	736	3.68
	4	805	4.03
	5	768	3.84
	6	775	3.88
	7	773	3.87
	8	734	3.67
	9	755	3.78
	10	747	3.74
	11	739	3.70
	12	761	3.81
	13	792	3.96
	14	783	3.92
	15	753	3.77
	16	803	4.01
	17	783	3.92
	18	758	3.79
	19	748	3.74
	20	796	3.98
	21	813	4.07
	22	764	3.82
	23	752	3.76
	24	787	3.94
	25	786	3.93
	26	734	3.67

Appendix B

AUC Computation

This appendix presents an explanation of the Area Under the receiver operating Curve (AUC) computation together with two detailed examples. According to Hand and Till [46], the AUC for a multi-class classifier can be estimated, as follows:

$$AUC = \frac{2}{c(c-1)} \sum_{i<j} A(i,j) \tag{B.1}$$

Where c is the number of class labels, i and j are classes numbers, and A is calculated as follows:

$$A(i,j) = \frac{MWW(i|j) + MWW(j|i)}{2} \tag{B.2}$$

Where MWW is the Man-Whitney-Wilcoxon statistic (or rank sum). This is calculated by first drawing up a MWW ranked table comprising two columns. The first column is the *response* (R) column and the second is the *signal* (S) column. The table comprises N rows, where N is the number of examples to be considered with respect to MWW calculation. Based on the signal and response values, each row in the table is given a *rank*. The ranking procedure is as follows (in descending order): true positives ($R_i = 1, S_i = 1$), false negatives ($R_i = 1, S_i = 0$), true negatives ($R_i = 0, S_i = 0$), and false positives ($R_i = 0, S_i = 1$). MWW is calculated as follows:

$$MWW(i,j) = \frac{s - \frac{n_1(n_1+1)}{2}}{n_1 n_2} \tag{B.3}$$

Where s is the sum of the rankings of (1s) in the signal column (sum of the rankings of positives) and n_1 and n_2 are the numbers of 1s (number of positives) and 0s (number of negatives) in the signal column.

The rest of this appendix presents two examples of AUC calculation.

B.1 Example One (100% Accurate Classifier)

This example presents the AUC calculation for a data set that feature three classes. The test data set (Truth Values) is presented in Table B.1, while the prediction values are presented in Table B.2. Note that the accuracy in this case is 100%, thus the classifier has correctly classified all the instances.

Example Num.	c1	c2	c3
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	1	0
6	0	0	1
7	0	0	1
8	0	0	1

TABLE B.1: Truth values

Example Num.	c1	c2	c3
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	1	0
6	0	0	1
7	0	0	1
8	0	0	1

TABLE B.2: Prediction values

In order to calculate the AUC for this classifier, MWW tables should be generated first for all possible pair-wise permutations of the classes: MWW(1|2), MWW(2|1), MWW(1|3), MWW(3|1), MWW(2|3) and MWW(3|2). Considering MWW(1|2), Table B.3 presents the corresponding MWW table. The table only considers the examples that should be classified as c1 or c2. Three examples were classified as c1 and two examples as not c1. The signal and response vectors are the same $\{0, 0, 1, 1, 1\}$ because the classifier was 100% accurate. With respect to MWW(1|2), $n_1 = 3, n_2 = 2, s = 3 + 4 + 5 = 12$. Thus: MWW(1|2)=$\frac{12-\frac{3(3+1)}{2}}{3*2}$ =1. With respect to MWW(2|1), the MWW table is presented in Table B.4. In this case $n_1 = 2, n_2 = 3$, and $s = 4 + 5 = 9$. Thus: MWW(2|1)=$\frac{9-\frac{2(2+1)}{2}}{2*3}$ =1. A(1,2) is then: A(1,2)=$\frac{1+1}{2}$=1.

Rank	Response	Signal
1	0	0
2	0	0
3	1	1
4	1	1
5	1	1

TABLE B.3: MWW(1|2)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	1	1
5	1	1

TABLE B.4: MWW(2|1)

Calculating MWW(1|3) as per Table B.5: MWW(1|3)= $\frac{15-\frac{3(3+1)}{2}}{3*3}$ =1. MWW(3|1)= $\frac{15-\frac{3(3+1)}{2}}{3*3}$ =1 (Table B.6). A(1,3) is then 1. Doing the same for MWW(2|3) and MWW(3|2) (Tables B.7 and B.8): MWW(2|3)=1 and MWW(3|2)=1. Thus A(2,3)=1. The AUC in this case: AUC = $\frac{2}{3(3-1)}(1 + 1 + 1) = 1$

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	1	1
5	1	1
6	1	1

Table B.5: MWW(1|3)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	1	1
5	1	1
6	1	1

Table B.6: MWW(3|1)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	1	1
5	1	1

Table B.7: MWW(2|3)

Rank	Response	Signal
1	0	0
2	0	0
3	1	1
4	1	1
5	1	1

Table B.8: MWW(3|2)

B.2 Example Two (Highly Unbalanced Data Set and One Class Does not Appear in the Test set)

The example presented in this section explains the issue mentioned earlier in Chapter 3 where a low AUC values is produced using TCV with respect to a *highly* unbalanced data sets. More specifically, assuming a highly unbalanced data set that feature less than ten examples of a specific class, dividing the data set into ten folds results in some folds without any instance from that class. During the testing stage, the classifier will not be actually evaluated against that class for some test folds (the folds that do not include any instance of that class). However the AUC calculations assume the complete number of class labels. The example presented in this section considers a data set that features three classes; however one class (c3) does not appear in the considered test fold. The test data set (in terms of Truth Values) is presented in Table B.9, while the prediction values are presented in Table B.10. Note that the accuracy in this case is 100%; the classifier correctly classified all instances.

Example Num.	c1	c2	c3
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	0	1	0
6	0	1	0
7	0	1	0
8	0	1	0

Table B.9: Truth values

Example Num.	c1	c2	c3
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	0	1	0
6	0	1	0
7	0	1	0
8	0	1	0

Table B.10: Prediction values

The MWW tables for all possible pair-wise permutations of the classes: MWW(1|2), MWW(2|1), MWW(1|3), MWW(3|1), MWW(2|3) and MWW(3|2) are presented in Tables B.11, B.12, B.13, B.14, B.15 and B.16.

Calculating MWW(1|2) as per Table B.11, MWW(1|2) $= \frac{26-\frac{4(4+1)}{2}}{4*4}$ =1. And MWW(2|1) as per Table B.12, MWW(2|1)= $\frac{26-\frac{4(4+1)}{2}}{4*4}$ =1. A(1,2) is then 1. MWW(1|3) = 0 because n_2=0, and MWW(3|1)= 0 because n_1=0 (see Tables B.13 and B.14). Then A(1,3)=0. A(2,3) is also 0 see Tables B.15, and B.16. The AUC in this case: AUC = $\frac{2}{3(3-1)}(1+0+0) = 0.33$

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	0	0
5	1	1
6	1	1
7	1	1
8	1	1

Table B.11: MWW(1|2)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	0	0
5	1	1
6	1	1
7	1	1
8	1	1

Table B.12: MWW(2|1)

Rank	Response	Signal
1	1	1
2	1	1
3	1	1
4	1	1

Table B.13: MWW(1|3)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	0	0

Table B.14: MWW(3|1)

Rank	Response	Signal
1	1	1
2	1	1
3	1	1
4	1	1

Table B.15: MWW(2|3)

Rank	Response	Signal
1	0	0
2	0	0
3	0	0
4	0	0

Table B.16: MWW(3|2)

Appendix C

Statistical Comparisons

C.1 Introduction

This appendix presents the detailes of the statistical comparison of the different structures, strategies, techniques and mechanisms that have been considered in this thesis with respect to the evaluation results presented earlier in Chapters 3, 4, 5 and 6. The broad aim is to determine firstly whether the results presented in these earlier chapters were indeed statistically significant and not purely a matter of chance and secondly to conduct a comparison between the different approaches proposed. To remind the reader of the different elements of the proposed techniques presented in this thesis Figure 1.1 from Chapter 1 is given again in Figure C.1.

The rest of this appendix is organised as follows: Sections C.2 and C.3 present the comparisons with respect to the Binary Tree and DAG hierarchical classification models respectively. Section C.4 presents a comparison between DAG based hierarchical classification and Binary Tree based hierarchical classification. Section C.5 then provides a comparisons between the suggested hierarchical classification model and alternative conventional ensemble classification models. Finally, a summary of the chapter is presented in Section C.6.

C.2 Binary Tree Hierarchical Classification Model Statistical Evaluation

This section reports on the statistical comparisons of the different classifiers, techniques, strategies, and mechanisms that were used with respect to the Binary Tree hierarchical classification model presented in Chapter 3. More specifically, the reported comparisons were directed at:

1. The different classifier generators (Decision tree, Naive Bayes, and CARM).
2. The different data distribution techniques (k-means clustering, *divisive* hierarchical clustering, and data splitting technique).

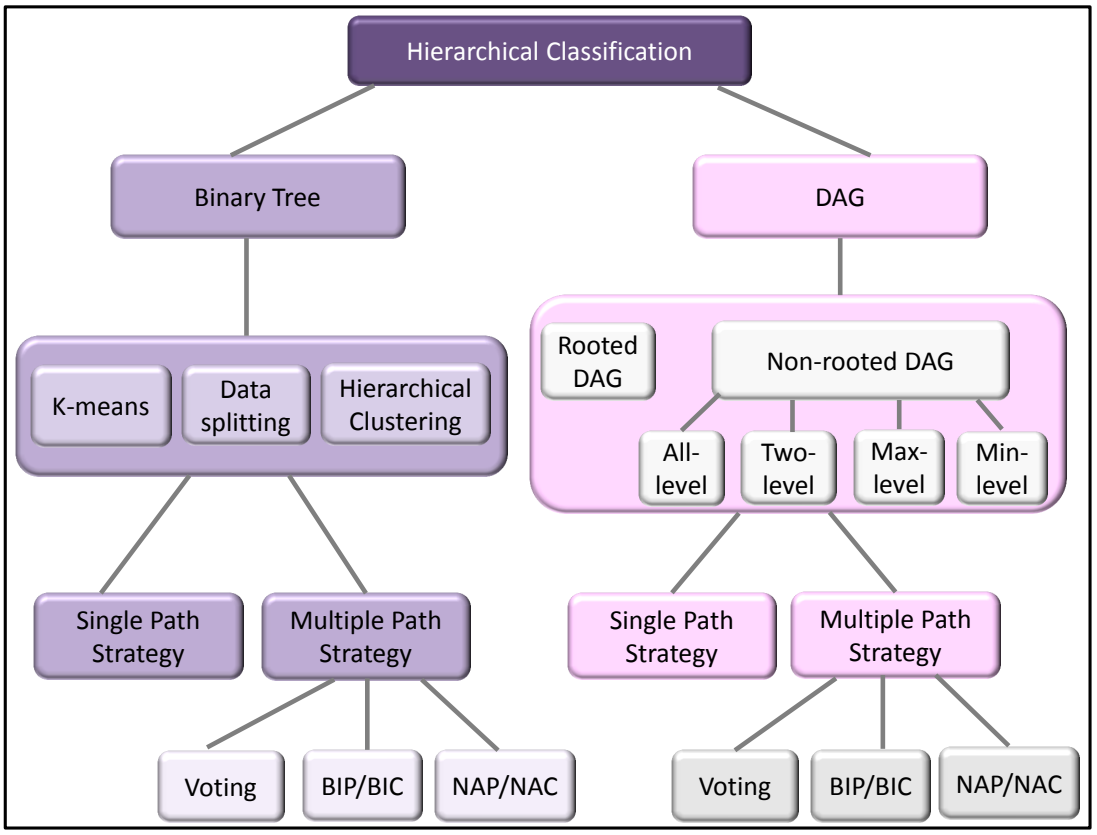


FIGURE C.1: Hierarchical classification structures, strategies, techniques, and mechanisms

3. The two classification strategies (Single and Multiple Path).

4. The three alternative mechanisms for arriving at a final classification decision in context of the Multiple Path strategy (Voting, Best Individual Probability BIP (or Confidence BIC) and Normalised Accumulated Probability NAP (or Confidence NAC)).

Thus a similar set of objectives to those reported on in Chapter 3 with respect to the binary tree hierarchical classification model. In the context of the above, the rest of this section is organised as follows:

1. Sub-section C.2.1 presents a comparison between the different classifiers considered, as well as a comparison of the different distribution techniques with respect to the binary tree hierarchical classification model when a Single Path strategy is adopted.

2. Sub-section C.2.2 presents a comparison between the different classifiers considered, as well as a comparison of the different distribution techniques and the two alternative mechanisms for arriving at a final classification decision (BIP/BIC and NAP/NAC) with respect to the binary tree hierarchical classification model when a Multiple Path strategy is adopted.

3. Sub-section C.2.3 presents a comparison between the Single and Multiple Path strategies.

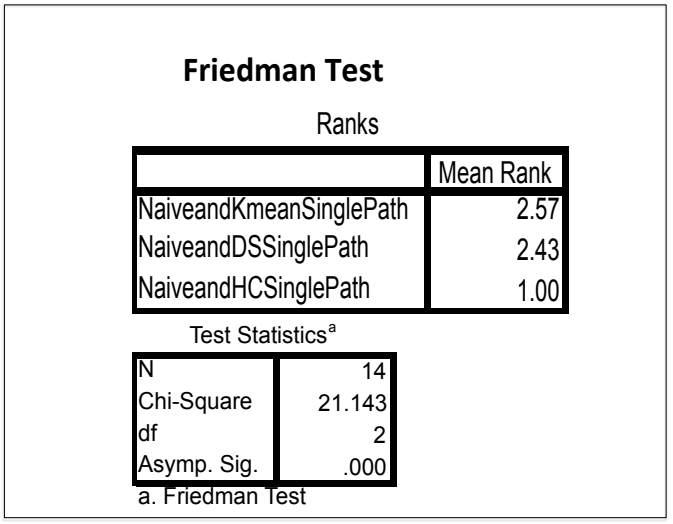
C.2.1 Comparative Evaluation using the Binary Tree Hierarchical Classification Model with the Single Path Strategy

As reported earlier in Chapter 3, the choice of: (i) the base classifiers and (ii) the data distribution technique, to be used to generate the Binary Tree hierarchical classification model, can affect the classification accuracy of the resulting ensemble model. From the evaluation presented in Chapter 3 it was found that the most effective classifier, to generate the binary tree hierarchical classification model with respect to the Single Path strategy, was found to be the Naive Bayes classifier. Regarding the data distribution technique, it was found that k-means and data splitting outperformed hierarchical clustering. The aim of this section is to determine whether this result was indeed statistically significant.

Commencing with the different data distribution techniques three statistical comparisons were performed; one with respect to each of the three considered classifiers. Regarding the comparison of the three considered data distribution techniques with respect to Naive Bayes classification, because this was a three-way comparison, the Friedman test was applied. Figure C.2 presents the results obtained. The results indicated that there was a significant difference between the three considered data distribution techniques ($X^2(2) = 21.143$, $p = 0.000$[1]). Since the Friedman test demonstrated a significant difference in performance between the considered distribution techniques, a post hoc test was applied to determine which classifier(s) performed significantly better than the others. As noted above, for this purpose the Nemenyi post-hoc test was used. First the critical difference was calculated according to Equation 2.2. $CD = 2.344\sqrt{\frac{3(3+1)}{6*14}} = 0.886$ (Note that α=0.05 was used). Then the difference between the average ranks was calculated for each pair of techniques and compared with the value of the critical difference. The results indicated that k-means and data splitting performed statistically better than hierarchical clustering (2.57 - 1.00 = 1.57 > 0.886, and 2.43 -1.00 = 1.43 > 0.886). While the performance of k-mean and data splitting was not statistically different (2.57 - 2.43= 0.14 < 0.886).

Figure C.3 presents the results of the Friedman test used to compare the operation of the three considered data distribution techniques with respect to Decision tree classification. The results indicated that there was a statistically significant difference between the three considered techniques ($X^2(2) = 6.627$, $p = 0.036$), hence the Nemenyi post-hoc test was applied. However, the Nemenyi test failed to detect any significant differences between the considered models (2.29 - 2.25 = 0.04 < 0.886, 2.29 - 1.46 = 0.83 < 0.886,

[1]The p-value is the estimated probability of rejecting the null hypothesis. If the p-value is less than or equal to the significance level (α) the null hypothesis can be rejected. α is the significance level of the statistical test, conventionally 5% or 1% [76].



Friedman Test

Ranks

	Mean Rank
NaiveandKmeanSinglePath	2.57
NaiveandDSSinglePath	2.43
NaiveandHCSinglePath	1.00

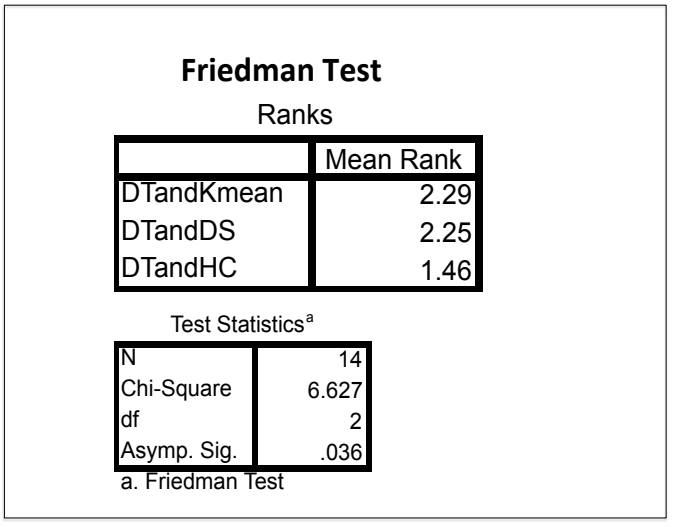
Test Statistics[a]

N	14
Chi-Square	21.143
df	2
Asymp. Sig.	.000

a. Friedman Test

FIGURE C.2: Results of Friedman test used to compare the different considered data distribution techniques with respect to Naive Bayes classification and the Single Path strategy

and 2.25 - 1.46 = 0.79 < 0.886). Recall from the above, and according to Demsar, that this is because of a recognised weakness of the post-hoc test.



Friedman Test

Ranks

	Mean Rank
DTandKmean	2.29
DTandDS	2.25
DTandHC	1.46

Test Statistics[a]

N	14
Chi-Square	6.627
df	2
Asymp. Sig.	.036

a. Friedman Test

FIGURE C.3: Results of Friedman test used to compare the different considered data distribution techniques with respect to Decision tree classification and Single Path strategy

Figure C.4 presents the results of the Friedman test used to compare the three considered data distribution techniques with respect to CARM classification. The results indicated that there was no statistically significant difference between the three considered data distribution techniques ($X^2(2) = 3.593$, $p = 0.166$).

Based on the above reported tests, and following on from the results presented in Chapter 3, it was thus concluded that there is a statistically significant difference in

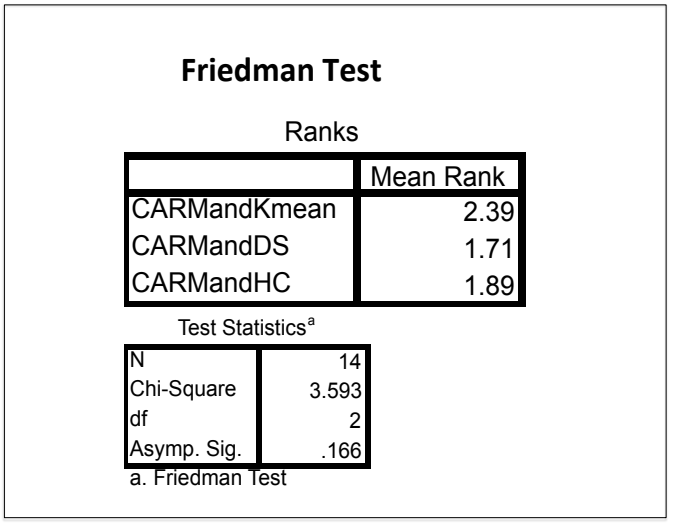


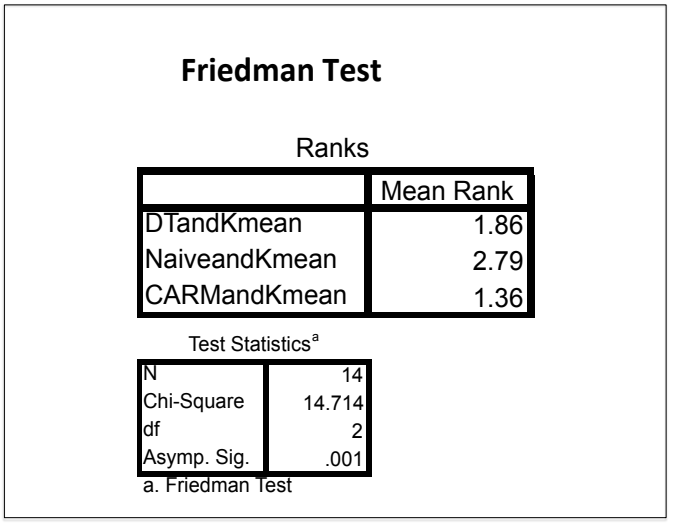
FIGURE C.4: Results of Friedman test used to compare the different considered data distribution techniques with respect to CARM classifier and Single Path strategy

operation between the three considered data distribution techniques, and that the k-means and the data splitting technique are more effective than hierarchical clustering. Consequently, a comparison of the different classifiers, considered to generate the Binary Tree hierarchical classification model, was conducted with respect to only the k-means and data splitting technique. Because we have three classifiers (Decision tree, Naive Bayes, and CARM) the Friedman test was again used.

Figure C.5 presents the results of the Friedman test used to compare the operation of the three considered classifiers with respect to k-means data distribution. According to the conducted Friedman test there was a statistically significant difference between the three considered classifiers ($X^2(2) = 14.714$, $p = 0.001$), hence the Nemenyi post-hoc test was applied. The result of the post-hoc test indicated that the operation of Naive Bayes was significantly better than when using either CARM (2.79 - 1.36 = 1.43 > 0.886) or Decision tree classification (2.79 -1.86 = 0.93 > 0.886). While the performances of Decision tree and CARM was not found to be significantly different (1.86 - 1.36 = 0.50 < 0.886).

Figure C.6 presents the results of Friedman test used to compare the three considered classifiers with respect to the data splitting technique. From the figure it can be seen that there was a statistically significant difference in the operation of the three considered classifiers ($X^2(2) = 14.286$, $p = 0.001$) and hence a Nemenyi post-hoc test was conducted. According to the Nemenyi test, Naive Bayes classification is again significantly more effective than CARM (2.71 - 1.29 = 1.42 > 0.886), while no statistically significant difference was observed between Naive Bayes and Decision tree classification (2.71 - 2.00 = 0.71 < 0.886) or between Decision tree and CARM classification (2.00 - 1.29 = 0.71 < 0.886).

From the foregoing, it was thus concludes that the most effective classifier with which to generate Binary Tree hierarchies was found to be Naive Bayes. Regarding the data



Friedman Test

Ranks

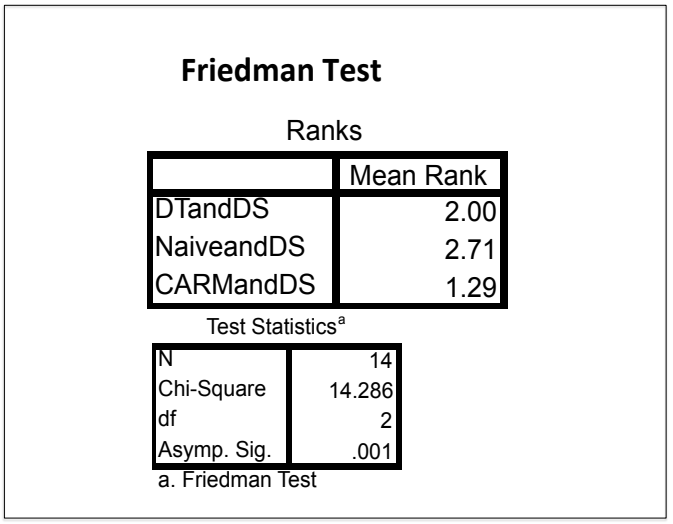
	Mean Rank
DTandKmean	1.86
NaiveandKmean	2.79
CARMandKmean	1.36

Test Statistics[a]

N	14
Chi-Square	14.714
df	2
Asymp. Sig.	.001

a. Friedman Test

FIGURE C.5: Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) utilised to generate the Binary Tree classification model with respect to k-means data distribution technique and Single Path strategy



Friedman Test

Ranks

	Mean Rank
DTandDS	2.00
NaiveandDS	2.71
CARMandDS	1.29

Test Statistics[a]

N	14
Chi-Square	14.286
df	2
Asymp. Sig.	.001

a. Friedman Test

FIGURE C.6: Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) utilised to generate the Binary Tree classification model with respect to data splitting technique and Single Path strategy

distribution technique, k-means and data splitting were found to be more effective than hierarchical clustering (with respect to the Single Path Strategy).

C.2.2 Comparative Evaluation using the Binary Tree Hierarchical Classification Model with the Multiple Path Strategy

Recall from Chapter 3 that the evaluation included in this chapter indicated that the most effective classifier, to generate the binary tree hierarchical classification model with respect to the Multiple Path strategy, was found to be the Naive Bayes classifier. Regarding the data distribution technique, it was found that k-means and data splitting

outperformed hierarchical clustering. The aim of the evaluation presented in this subsection is to determine whether this result was statistically significant or not. This section also reports on the comparison undertaken with respect to the three alternative mechanisms that were considered to arrive at a final classification decision (Voting, BIP/BIC, and NAP/NAC).

Commencing with the statistical evaluation of the results obtained to determine the effectiveness of the different data distribution techniques two sets of experiments were conducted; one with respect to Naive Bayes and one with respect to CARM (recall that decision tree classification is not appropriate in the context of the Multiple Path Strategy). Figure C.7 presents the results of the Friedman test used for the first set of experiments. The presented results indicated that there was a statistically significant difference in operation using the three considered data distribution techniques ($X^2(2) = 18.143$, $p = 0.000$). A consequent Nemenyi post-hoc test was thus applied from which it was observed that k-means and data splitting performed statistically better than hierarchical clustering (2.50 - 1.07 = 1.43 > 0.886, and 2.43 -1.07 = 1.36 > 0.886). While the performance of k-means and data splitting was not statistically different (2.50 - 2.43= 0.07 < 0.886).

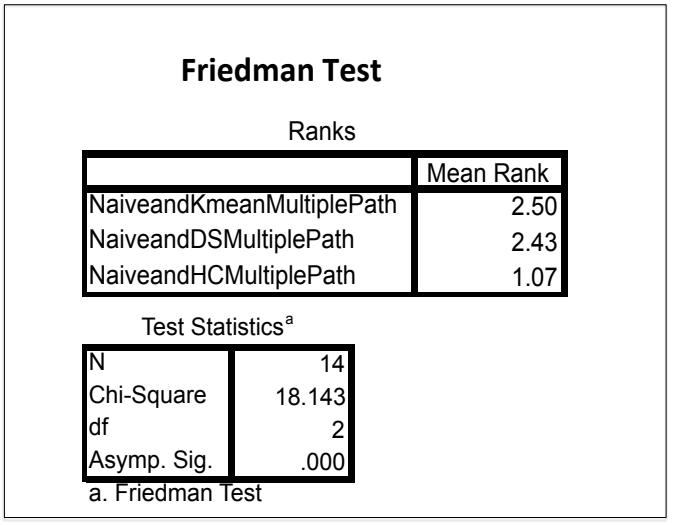


FIGURE C.7: Results of Friedman test used to compare the different data distribution techniques with respect to Naive Bayes classification and the Multiple Path strategy

Regarding the comparison of the different data distribution techniques with respect to CARM classification when coupled with the Multiple Path strategy Figure C.8 presents the Friedman test results obtained. From the figure it can be seen that there was a statistically significant difference between the three considered data distribution techniques ($X^2(2) = 7.018$, $p = 0.030$). The consequent Nemenyi test indicated that k-means distribution was more effective than data splitting (2.57 - 1.68 = 0.89 > 0.886). While the performance difference between k-means and hierarchical clustering was found to not be statistically different (2.57 - 1.75= 0.82 < 0.886). Similarly no statistically significant performance difference was found between the data splitting and

hierarchical clustering distribution techniques (1.75 - 1.68 = 0.07 < 0.886). Again, recall that an issue with using confidence values, generated when using CARM, to determine whether one or two branches emanating from a node should be followed, as reported in Chapter 3, was that it was not always possible to identify both branch confidence values for a given node. Consequently the unknown confidence values affected the results of Multiple Path strategy. This issue would affect the outcome of any comparison of the different data distribution techniques coupled with CARM classification and the Multiple Path strategy.

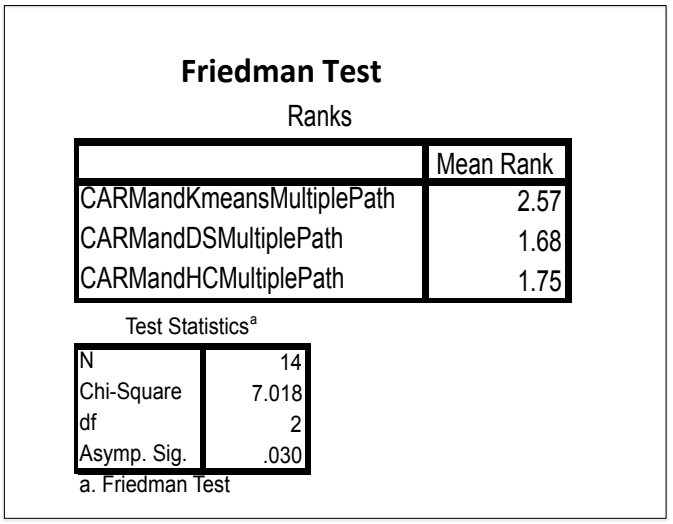


FIGURE C.8: Results of Friedman test used to compare the different data distribution techniques with respect to CARM classification and the Multiple Path strategy

Although, from the foregoing, it is clear that Naive Bayes classification is more statistically effective than CARM classification, with respect to the Multiple Path strategy, a comparison is presented in Figure C.9 with respect to the k-mean and data splitting technique. Because the goal here is to statistically compare two classification models, the Wilcoxon test was applied. According to the outcome of the Wilcoxon signed rank test, there was a significant difference in performance between the two models; Naive Bayes classification was found to be statistically more effective than CARM classification, regardless of the adopted data distribution technique. With respect to k-means z = -3.300 and p < 0.05, and regarding data splitting z = -3.236 and p < 0.05.

Regarding the comparison between the different mechanisms (Voting, BIP, and NAP) for arriving at a final classification decision the Friedman test was applied (because the goal here was to compare three models). Commencing with the comparison of Voting, BIP, and NAP with respect to data splitting technique. Figure C.10 presents the obtained results. From the figure it can be seen that there was a statistical difference between the three considered mechanisms ($X^2(2) = 6.151$, $p = 0.046$). The consequent Nemenyi test indicated that the NAP mechanism was more effective than Voting (2.39 - 1.50 = 0.89 > 0.886). While the performance difference between NAP and BIP was found to not be significantly different (2.39 - 2.11= 0.28 < 0.886). Similarly no significant

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
CARMandKmeanMultiplePath - NaiveandKmeanMultiplePath	Negative Ranks	14[a]	7.50	105.00
	Positive Ranks	0[b]	0.00	0.00
	Ties	0[c]		
	Total	14		
CARMandDSMultiplePath - NaiveandDSMultiplePath	Negative Ranks	13[d]	8.00	104.00
	Positive Ranks	1[e]	1.00	1.00
	Ties	0[f]		
	Total	14		

a. CARMandKmeanMultiplePath < NaiveandKmeanMultiplePath
b. CARMandKmeanMultiplePath > NaiveandKmeanMultiplePath
c. CARMandKmeanMultiplePath = NaiveandKmeanMultiplePath
d. CARMandDSMultiplePath < NaiveandDSMultiplePath
e. CARMandDSMultiplePath > NaiveandDSMultiplePath
f. CARMandDSMultiplePath = NaiveandDSMultiplePath

Test Statistics[a]

	CARMandKmeanMultiplePath - NaiveandKmeanMultiplePath	CARMandDSMultiplePath - NaiveandDSMultiplePath
Z	-3.300[b]	-3.236[b]
Asymp. Sig. (2-tailed)	.001	.001

a. Wilcoxon Signed Ranks Test
b. Based on positive ranks.

FIGURE C.9: Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to the different data distribution techniques and the Multiple Path strategy

performance difference was found between the BIP and Voting techniques (2.11 - 1.50 $= 0.61 < 0.886$).

Friedman Test

Ranks

	Mean Rank
NaiveandDSMultiPathWithNAP	2.39
NaiveandDSMultipathWithBIP	2.11
NaiveandDSMultiPathWithVoting	1.50

Test Statistics[a]

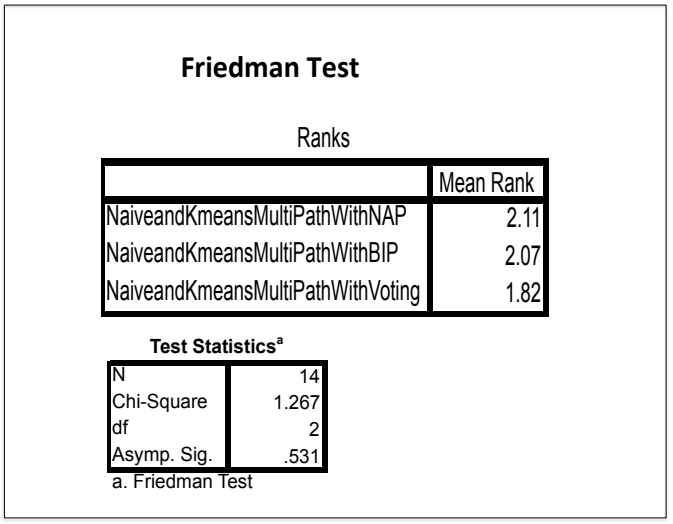
N	14
Chi-Square	6.151
df	2
Asymp. Sig.	.046

a. Friedman Test

FIGURE C.10: Results of Friedman test used to compare the three alternative mechanisms to arrive at a final classification decision with respect to the Multiple Path strategy: (i) Voting, (ii) BIP, and (iii) NAP with respect to data splitting mechanism

Figure C.11 presents the obtained results of applying Friedman test to compare the three alternative mechanisms (Voting, BIP, and NAP) with respect to k-means.

Analysis of the test results indicated that no statistically significant difference was found in performance between the three considered mechanisms ($X^2(2) = 1.267$, $p = 0.531$).



Friedman Test

Ranks

	Mean Rank
NaiveandKmeansMultiPathWithNAP	2.11
NaiveandKmeansMultiPathWithBIP	2.07
NaiveandKmeansMultiPathWithVoting	1.82

Test Statistics[a]

N	14
Chi-Square	1.267
df	2
Asymp. Sig.	.531

a. Friedman Test

FIGURE C.11: Results of Friedman test used to compare the three alternative mechanisms to arrive at a final classification decision with respect to the Multiple Path strategy: (i) Voting, (ii) BIP, and (iii) NAP with respect to K-means data distribution technique

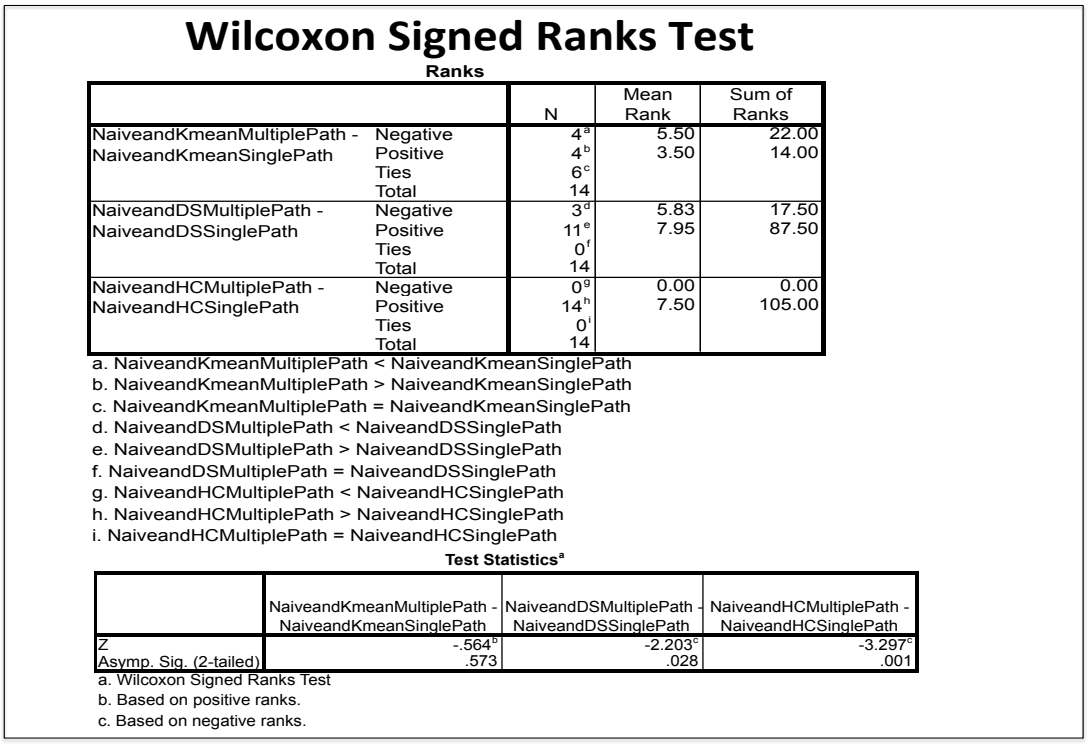
From the foregoing we can conclude: (i) as in the case of the Single Path strategy, that Naive Bayes classification was more effective than CARM classification when used to generate a Binary Tree hierarchical classification model with respect to the Multiple Path strategy, (ii) the k-means and data splitting techniques are more effective than hierarchical clustering (again as also noted in the case of Single Path strategy), and (iii) the NAP mechanism statistically outperformed the Voting mechanism for arriving at a final classification decision with respect to the Multiple Path strategy. In addition, no statistically significant difference was detected in the operation of the BIP and NAP mechanisms.

C.2.3 Comparison between Single and Multiple Path Strategies when using The Binary Tree Hierarchical Classification Model

Because of (i) earlier experiments reported in Chapter 3 (and confirmed in the foregoing section) had indicated that Naive Bayes classification was the most effective classifier with which to generate the desired Binary Tree hierarchical classification model, and (ii) the issues that were reported in Chapter 3 regarding following multiple paths with respect to CARM classifiers, this section presents the statistical comparison between the Single and Multiple path strategies with respect to Naive Bayes classification only and the three considered data distribution techniques (k-mean, data splitting, and *divisive* hierarchical clustering).

The Wilcoxon test was applied because the goal here was to compare the two strategies, Single and Multiple Path. Figure C.12 presents the results obtained. From the

figure it can be seen that there was a statistical performance difference between the two strategies with respect to data splitting and hierarchical clustering, z = -2.203 and p = 0.028, and z = -3.297 and p = 0.001 ($p < 0.05$) respectively. While there was no significant difference between the two strategies with respect to k-means, z = -0.564 and p = 0.573. These results support what was argued earlier in chapter 3, that the Multiple Path strategy avoids many of the mis-classifications that occur when using the Single Path strategy.



Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
NaiveandKmeanMultiplePath - NaiveandKmeanSinglePath	Negative	4[a]	5.50	22.00
	Positive	4[b]	3.50	14.00
	Ties	6[c]		
	Total	14		
NaiveandDSMultiplePath - NaiveandDSSinglePath	Negative	3[d]	5.83	17.50
	Positive	11[e]	7.95	87.50
	Ties	0[f]		
	Total	14		
NaiveandHCMultiplePath - NaiveandHCSinglePath	Negative	0[g]	0.00	0.00
	Positive	14[h]	7.50	105.00
	Ties	0[i]		
	Total	14		

a. NaiveandKmeanMultiplePath < NaiveandKmeanSinglePath
b. NaiveandKmeanMultiplePath > NaiveandKmeanSinglePath
c. NaiveandKmeanMultiplePath = NaiveandKmeanSinglePath
d. NaiveandDSMultiplePath < NaiveandDSSinglePath
e. NaiveandDSMultiplePath > NaiveandDSSinglePath
f. NaiveandDSMultiplePath = NaiveandDSSinglePath
g. NaiveandHCMultiplePath < NaiveandHCSinglePath
h. NaiveandHCMultiplePath > NaiveandHCSinglePath
i. NaiveandHCMultiplePath = NaiveandHCSinglePath

Test Statistics[a]

	NaiveandKmeanMultiplePath - NaiveandKmeanSinglePath	NaiveandDSMultiplePath - NaiveandDSSinglePath	NaiveandHCMultiplePath - NaiveandHCSinglePath
Z	-.564[b]	-2.203[c]	-3.297[c]
Asymp. Sig. (2-tailed)	.573	.028	.001

a. Wilcoxon Signed Ranks Test
b. Based on positive ranks.
c. Based on negative ranks.

FIGURE C.12: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the three considered data distribution techniques

C.3 DAG Classification Models Evaluation

In this section a statistical comparisons of the different approaches, strategies, and mechanisms with respect to the proposed DAG classification models are presented. Recall that the aim is to determine whether the results from earlier experiments, reported in Chapters 4, 5, and 6, were indeed statistically significant and to compare the results obtained. The section is organised as follows. Sub-section C.3.1 presents the statistical analysis of the outcomes from the set of experiments designed to evaluate the DAG classification model when a Single Path strategy was adopted; while Sub-section C.3.2 presents the statistical analysis of the results from a similar set of experiments designed to evaluate the DAG classification model when a Multiple Path strategy was adopted. Sub-section C.3.3 then presents a comparison between the two strategies (Single and

Multiple Path), and Sub-section C.3.4 presents a comparison between the different DAG models considered in this thesis.

C.3.1 Comparative Evaluation using the DAG Classification Models with the Single Path Strategy

In this sub-section the statistical comparison between the operation of the different classifiers (Decision tree, Naive Bayes, and CARM) utilised to generate the different DAG models (Rooted, All-level, and Two-level), with respect to the Single Path strategy, is presented. Note here that with respect to the DAG classification model coupled with breadth pruning (Max-level and Min-level DAGs), only Naive Bayes classification was considered because of the issues reported in Chapter 5 with respect to CARM classifiers. In Chapters 4 and 5 it was established, based on experimentation that, the most effective classifier to generate the DAG classification model was found to be Naive Bayes classifier. The aim of this sub-section is to conduct a statistical evaluation of these results to determine if these results are indeed statistically significant and to conduct a general comparison to identify the best classifier.

Commencing with the *rooted* DAG classification model, Figure C.13 presents the results of Friedman test used to compare the three considered classifiers with respect to the *rooted* DAG classification model and the Single Path strategy. According to the conducted Friedman test there was a statistically significant difference in operation between the three considered classifiers ($X^2(2) = 11.091$, $p = 0.004$). The consequent Nemenyi post-hoc test, using $CD = 2.344\sqrt{\frac{3(3+1)}{6*11}} = 0.999$ demonstrated that Naive Bayes performed statistically better than both Decision tree and CARM classification (2.82 - 1.55 = 1.27 > 0.999, and 2.82 - 1.64 = 1.18 > 0.999); while no statistically significant difference was observed between Decision tree and CARM classification (1.64 - 1.55 = 0.09 < 0.999).

With respect to the *non-rooted* DAG classification models, only two classifiers were considered, Naive Bayes and CARM, for All-level and Two-level DAGs. Because the goal here was to compare the operation of the two classifiers with respect to the All-level and Two-level DAGs the Wilcoxon test was again applied. Figure C.14 presents the obtained results from the Wilcoxon tests. From the figure it can be seen that there is a statistically significant difference in operation between the two classifiers with respect to both the All-level and Two-level DAGs. Regarding the All-level DAG z = -2.253, p = 0.024 (p < 0.05); with respect to the Two-level DAG z = -2.312, p = 0.021 (p < 0.05). Again the results demonstrated that Naive Bayes classification was the most effective confirming the results from earlier experiments reported in Chapters 4 and 5.

C.3.2 Comparative Evaluation using the DAG Classification Models with the Multiple Path Strategy

This section describes the outcomes from the statistical evaluation conducted to compare the operation of the multiple path DAGs (*rooted* and *non-rooted*) with respect to: (i)

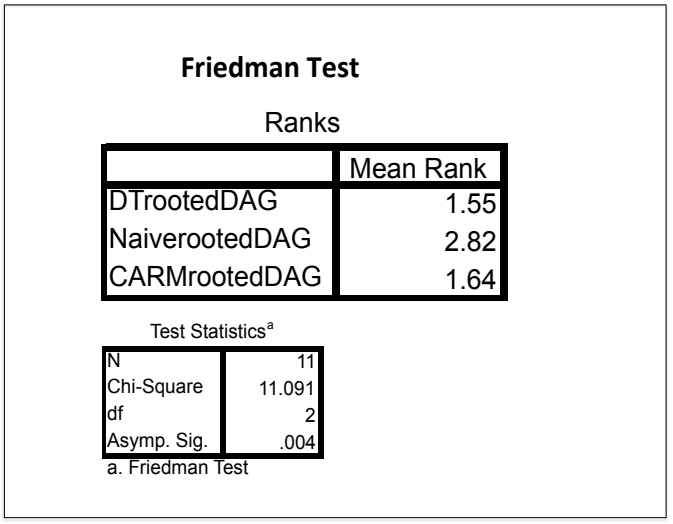


FIGURE C.13: Results of Friedman test used to compare the different classifiers (Decision tree, Naive Bayes, and CARM) with respect to the rooted DAG classification model and the Single Path strategy

the nature of the classifiers used at the nodes, and (ii) the effect of using Voting, BIP, and NAP to arrive at a final classification. From the evaluation presented in Chapters 4 and 5 it was found that the most effective classifier, to generate the DAG classification models with respect to the Multiple Path strategy, was found to be the Naive Bayes classifier. With respect to class label selection mechanism, it was found that the results of the three mechanisms are very similar. The aim of this section is to determine whether the obtained results were indeed statistically significant.

Commencing with the *rooted* DAG classification model. Because only two classifiers (Naive Bayes and CARM) were utilised a Wilcoxon test was applied. Figure C.15 presents the results obtained. From the figure it can be seen that a statistically significant difference between the two classifiers was identified, z = -2.936 and p = 0.003 ($p < 0.05$). Again Naive Bayes classification was found to be more effective than CARM classification with respect to the *rooted* DAG multiple path model (confirming the results from earlier experiments reported on in Chapter 4).

With respect to the *non-rooted* DAG multiple path models (All-level and Two-level), as in the above case, only two classifiers were considered, Naive Bayes and CARM, for each variation; and consequently Wilcoxon tests were again applied. The results are presented in Figure C.16 from which it can be seen that there was a statistically significant difference in operation between the two classifiers with respect to both DAG variations (All-level and Two-level). Regarding the All-level variation z = -2.199 and p = 0.028 ($p < 0.05$). With respect to the Two-level variation z = -2.429, and p = 0.015 ($p < 0.05$). Again Naive Bayes classification was identified as being the most effective in the case *non-rooted* DAG multiple path model (confirming the results from earlier experiments reported on in Chapter 5).

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
CARMAlllevelDAGSinglePath - NaiveallIevelDAGSinglePath	Negative	8[a]	5.19	41.50
	Positive	1[b]	3.50	3.50
	Ties	0[c]		
	Total	9		
CARMTwolevelDAGSinglePath - NaiveTwolevelDAGSinglePath	Negative	8[d]	5.25	42.00
	Positive	1[e]	3.00	3.00
	Ties	0[f]		
	Total	9		

a. CARMAlllevelDAGSinglePath < NaiveallIevelDAGSinglePath
b. CARMAlllevelDAGSinglePath > NaiveallIevelDAGSinglePath
c. CARMAlllevelDAGSinglePath = NaiveallIevelDAGSinglePath
d. CARMTwolevelDAGSinglePath < NaiveTwolevelDAGSinglePath
e. CARMTwolevelDAGSinglePath > NaiveTwolevelDAGSinglePath
f. CARMTwolevelDAGSinglePath = NaiveTwolevelDAGSinglePath

Test Statistics[a]

	CARMAlllevelDAGSinglePath - NaiveallIevelDAGSinglePath	CARMTwolevelDAGSinglePath - NaiveTwolevelDAGSinglePath
Z	-2.253[b]	-2.312[b]
Asymp. Sig. (2-tailed)	.024	.021

a. Wilcoxon Signed Ranks Test
b. Based on positive ranks.

FIGURE C.14: Results of Wilcoxon test used to compare the two considered classifiers with respect to: (i) All-level and (ii) Two-level DAGs, and the Single Path strategy

Regarding the comparison between the three final class allocation mechanisms, Voting, BIP, and NAP, for arriving at a final classification decision the Friedman test results are presented in Figure C.17. Inspection of the figure indicates that there was no significant difference in operation performance between the three mechanisms ($X^2(2) = 0.353$, $p = 0.838$), confirming earlier results.

C.3.3 Comparison between Single and Multiple Path Strategies using the DAG Classification Models

Recall from Chapters 4 and 5 (and from the foregoing sections) that Naive Bayes classification is the most effective classification to be employed when generating the DAGs, this section presents a statistical comparison between the Single and Multiple Path strategies with respect to Naive Bayes classification only.

Commencing with a comparison of the Single and Multiple Path strategies with respect to the *rooted* DAG model, Section 4.4.3 demonstrated that following more than one path within the *rooted* DAG classification model produced a better classification effectiveness with respect to some of the considered data sets. Figure C.18 presents the results obtained from a Wilcoxon test. The reported results show that there was no statistically significant difference in operation between the two strategies with respect to *rooted* DAG model, z = -0.447, and p = 0.655.

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
CARMrootedDAGMultiplePath - NaiverootedDAGMultiplePath	Negative	11[a]	6.00	66.00
	Positive	0[b]	0.00	0.00
	Ties	0[c]		
	Total	11		

a. CARMrootedDAGMultiplePath < NaiverootedDAGMultiplePath
b. CARMrootedDAGMultiplePath > NaiverootedDAGMultiplePath
c. CARMrootedDAGMultiplePath = NaiverootedDAGMultiplePath

Test Statistics[a]

	CARMrootedDAGMultiplePath - NaiverootedDAGMultiplePath
Z	-2.936[b]
Asymp. Sig. (2-tailed)	.003

a. Wilcoxon Signed Ranks Test
b. Based on positive ranks.

FIGURE C.15: Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to the rooted DAG multiple path model

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
CARMAlllevelDAGMultiplePath - NaiveAlllevelDAGMultiplePath	Negative	8[a]	5.13	41.00
	Positive	1[b]	4.00	4.00
	Ties	0[c]		
	Total	9		
CARMTwolevelDAGMultiplePath - NaiveTwolevelDAGMultiplePath	Negative	8[d]	5.38	43.00
	Positive	1[e]	2.00	2.00
	Ties	0[f]		
	Total	9		

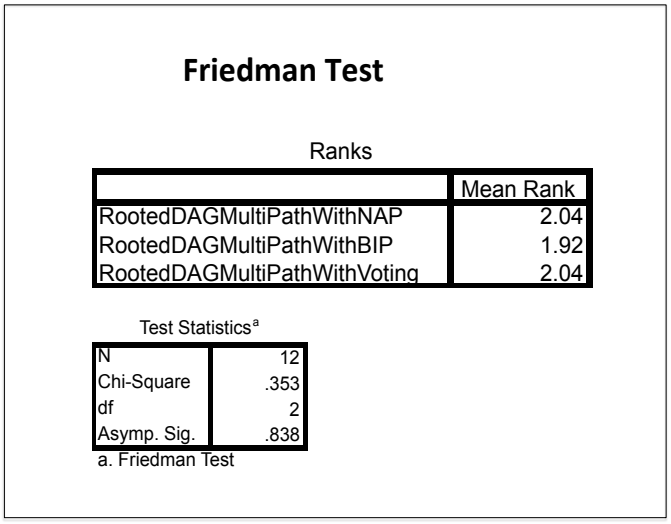
a. CARMAlllevelDAGMultiplePath < NaiveAlllevelDAGMultiplePath
b. CARMAlllevelDAGMultiplePath > NaiveAlllevelDAGMultiplePath
c. CARMAlllevelDAGMultiplePath = NaiveAlllevelDAGMultiplePath
d. CARMTwolevelDAGMultiplePath < NaiveTwolevelDAGMultiplePath
e. CARMTwolevelDAGMultiplePath > NaiveTwolevelDAGMultiplePath
f. CARMTwolevelDAGMultiplePath = NaiveTwolevelDAGMultiplePath

Test Statistics[a]

	CARMAlllevelDAGMultiplePath - NaiveAlllevelDAGMultiplePath	CARMTwolevelDAGMultiplePath - NaiveTwolevelDAGMultiplePath
Z	-2.199[b]	-2.429[b]
Asymp. Sig. (2-tailed)	.028	.015

a. Wilcoxon Signed Ranks Test
b. Based on positive ranks.

FIGURE C.16: Results of Wilcoxon test used to compare Naive Bayes and CARM classification with respect to: (i) All-level, and (ii) Two-level DAG multiple path model



Friedman Test

Ranks

	Mean Rank
RootedDAGMultiPathWithNAP	2.04
RootedDAGMultiPathWithBIP	1.92
RootedDAGMultiPathWithVoting	2.04

Test Statistics[a]

N	12
Chi-Square	.353
df	2
Asymp. Sig.	.838

a. Friedman Test

FIGURE C.17: Results of Fridman test used to compare the operation of the three alternative mechanisms used to arrive at a final classification when using the proposed Multiple Path strategies: (i) Voting, (ii) BIP, and (iii) NAP with respect to Naive Bayes classification and the rooted DAG approach

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
rootedDAGMultiplePathStrategy - rootedDAGSinglePathStrategy	Negative	1[a]	1.00	1.00
	Positive	1[b]	2.00	2.00
	Ties	10[c]		
	Total	12		

a. rootedDAGMultiplePathStrategy < rootedDAGSinglePathStrategy
b. rootedDAGMultiplePathStrategy > rootedDAGSinglePathStrategy
c. rootedDAGMultiplePathStrategy = rootedDAGSinglePathStrategy

Test Statistics[a]

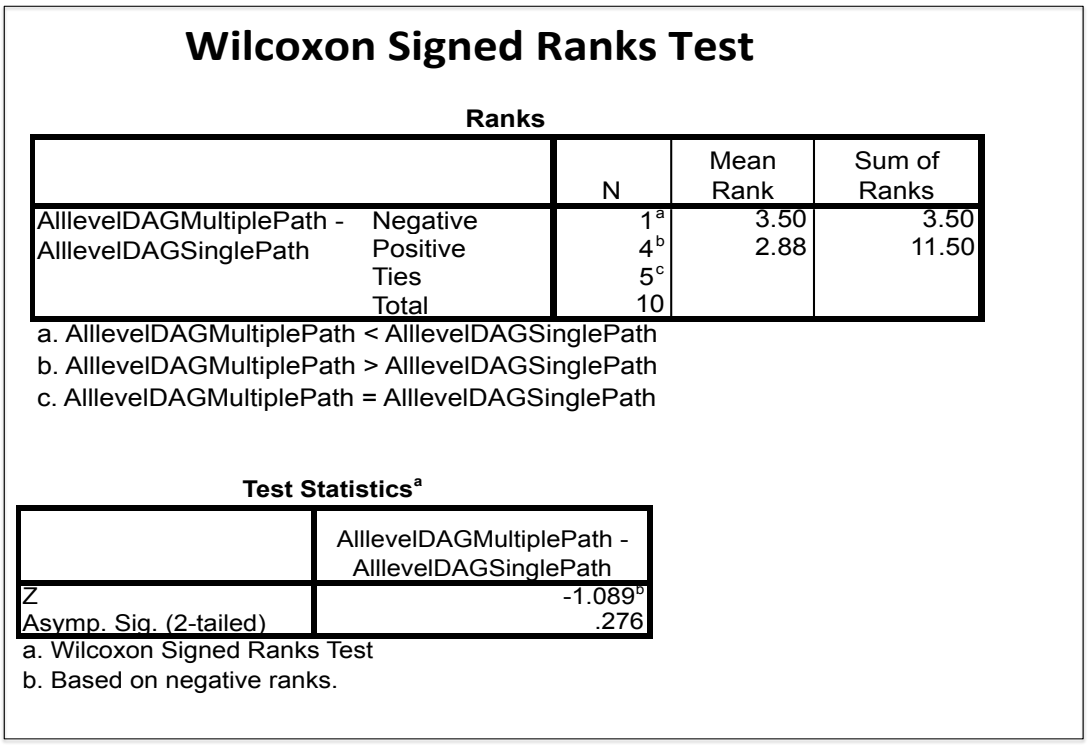
	rootedDAGMultiplePathStrategy - rootedDAGSinglePathStrategy
Z	-.447[b]
Asymp. Sig. (2-tailed)	.655

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.18: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the rooted DAG model

Regarding the comparison of the Single and Multiple Path strategies with respect the All-level DAG model, Section 5.4.3 reported that using the Multiple Path strategy a better classification accuracy could be obtained than when the Single Path strategy was adopted. Figure C.19 presents the results of the Wilcoxon test. According to this

test there was again no statistically significant difference in operation between the two strategies with respect to the All-level DAG model, z = -1.089, and p = 0.276.



Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
AlllevelDAGMultiplePath - AlllevelDAGSinglePath	Negative	1[a]	3.50	3.50
	Positive	4[b]	2.88	11.50
	Ties	5[c]		
	Total	10		

a. AlllevelDAGMultiplePath < AlllevelDAGSinglePath
b. AlllevelDAGMultiplePath > AlllevelDAGSinglePath
c. AlllevelDAGMultiplePath = AlllevelDAGSinglePath

Test Statistics[a]

	AlllevelDAGMultiplePath - AlllevelDAGSinglePath
Z	-1.089[b]
Asymp. Sig. (2-tailed)	.276

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.19: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the All-level DAG model

With respect to the statistical comparison between the Single and Multiple Path strategies in context of the Two-level DAG, Section 5.4.3 demonstrated that using the Multiple Path strategy a better classification accuracy could be obtained with respect to at least some of the considered data sets. Figure C.20 presents the Wilcoxon test results. According to the figure there was also no statistically significant difference in operation between the two strategies with respect to the Two-level DAG model, z = -0.184, and p = 0.854.

In the case of the comparison between the Single and Multiple Path strategies in context of Max-level DAG, Section 6.4.3 concluded that following multiple paths within the Max-level DAG tended to produce a better classification effectiveness, than when following only single paths. Figure C.21 presents the obtained Wilcoxon test results. According to results there was also no statistically significant operational difference between the two strategies with respect to the Max-level DAG model, z = -0.318, and p = 0.750.

Finally, when comparing the Single and Multiple Path strategies in context of Min-level DAG the Wilcoxon test results shown in Figure C.22 were obtained. From the figure it can be seen that again there was no statistically significant difference between the two strategies, z = -0.813, and p = 0.416. Recall that Section 6.4.3 demonstrated

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
TwolevelDAGMultiplePath - TwolevelDAGSinglePath	Negative	2[a]	2.25	4.50
	Positive	2[b]	2.75	5.50
	Ties	8[c]		
	Total	12		

a. TwolevelDAGMultiplePath < TwolevelDAGSinglePath
b. TwolevelDAGMultiplePath > TwolevelDAGSinglePath
c. TwolevelDAGMultiplePath = TwolevelDAGSinglePath

Test Statistics[a]

	TwolevelDAGMultiplePath - TwolevelDAGSinglePath
Z	-.184[b]
Asymp. Sig. (2-tailed)	.854

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.20: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Two-level DAG model

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
MaxlevelDAGMultiplePath - MaxlevelDAGSinglePath	Negative	2[a]	4.50	9.00
	Positive	4[b]	3.00	12.00
	Ties	4[c]		
	Total	10		

a. MaxlevelDAGMultiplePath < MaxlevelDAGSinglePath
b. MaxlevelDAGMultiplePath > MaxlevelDAGSinglePath
c. MaxlevelDAGMultiplePath = MaxlevelDAGSinglePath

Test Statistics[a]

	MaxlevelDAGMultiplePath - MaxlevelDAGSinglePath
Z	-.318[b]
Asymp. Sig. (2-tailed)	.750

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.21: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Max-level DAG model

that following multiple paths within the Min-level DAG produced a better classification effectiveness with respect to some of the considered data sets.

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
MinlevelDAGMultiplePath - MinlevelDAGSinglePath	Negative	2[a]	2.25	4.50
	Positive	3[b]	3.50	10.50
	Ties	7[c]		
	Total	12		

a. MinlevelDAGMultiplePath < MinlevelDAGSinglePath
b. MinlevelDAGMultiplePath > MinlevelDAGSinglePath
c. MinlevelDAGMultiplePath = MinlevelDAGSinglePath

Test Statistics[a]

	MinlevelDAGMultiplePath - MinlevelDAGSinglePath
Z	-.813[b]
Asymp. Sig. (2-tailed)	.416

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.22: Results of Wilcoxon test used to compare the Single and Multiple Path strategies with respect to Naive Bayes classification and the Min-level DAG model

From the foregoing we can confirm that, unlike in the case of the Binary Tree hierarchical classification model, following multiple paths within the DAG classification model did not produce a statistically significant difference in classification effectiveness than when following only a single path. The reason for this is that the combination technique used to distribute classes between nodes in the DAG resulted in well-defined class labels at each DAG node, unlike in the case where clustering algorithms were used with respect to the Binary Tree hierarchical classification model; consequently the number of mis-classifications is less and the effect of following multiple paths within the DAG is not highly significant.

C.3.4 Comparison Between the Different DAG Models

This section presents a comparison between the different DAG models considered in this thesis: (i) Rooted DAG, (ii) All-level DAG, (iii) Two-level DAG, (iv) Max-level DAG, and (v) Min-level DAG (recall that the last four can be grouped under the heading "non-rooted" DAG). The objective of the comparison reported on in this section was to determine the most effective DAG structure. Again only Naive Bayes classification was considered with respect to the classifiers held at the DAG nodes for reasons given earlier in this chapter.

Commencing with the statistical comparing for all the different DAG models when using the Single Path strategy the Friedman test results are presented in Figure C.23. From the figure it can be seen that there was no statistically significant operational difference between the different models ($X^2(2) = 2.015$, $p = 0.733$) in terms of classification effectiveness. Similarly, with respect to the statistical comparison of the considered DAG models with respect to the Multiple Path strategy the Friedman test results are given in Figure C.24. From the figure it can again be seen that there was no statistically significant difference between the different considered models ($X^2(2) = 3.848$, $p = 0.427$) in terms of classification effectiveness.

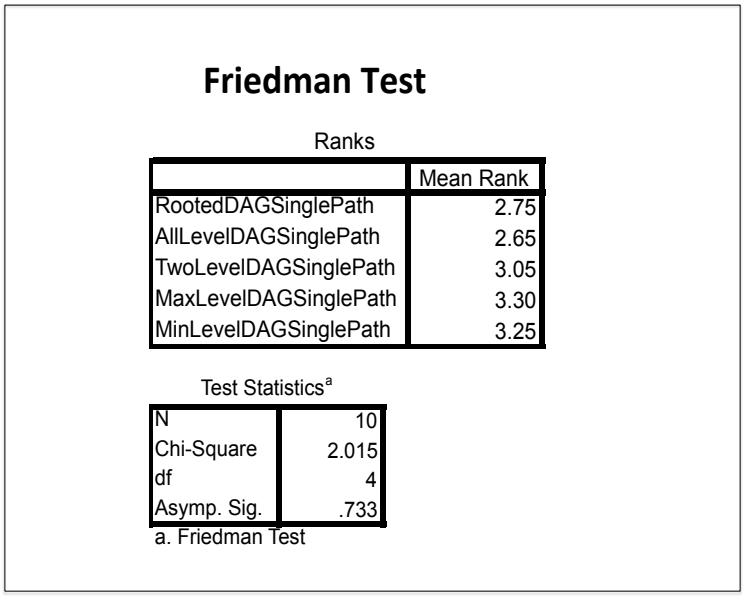
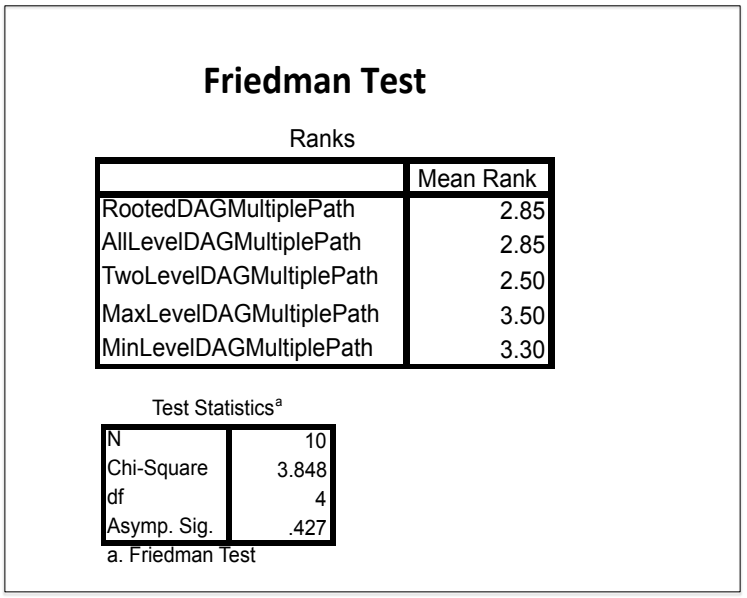


FIGURE C.23: Results of Friedman test used to compare the different DAG models with respect to the Single Path strategy

Thus we can conclude that the operation of the considered DAG models are not statistically different, regardless of the adopted strategy (Single or Multiple Path). Although it should be recalled that the Min-level DAG is the most efficient.

C.4 Comparison Between DAG Based Hierarchical Classification and Binary Tree Based Hierarchical Classification

This section presents a comparison between the two main models of hierarchical classification proposed in this thesis, the Binary Tree and DAG models, with respect to both the Single and Multiple Path strategies. With respect to the Binary Tree model Naive Bayes classification and data splitting was adopted for this purpose. While regarding the DAG classification model a Min-level DAG generated using Naive Bayes classifiers was adopted. The reason behind selecting these variations was that it had been previously established (Sections 3.4.2 and 6.4.4) that they generated the best results with respect



Friedman Test

Ranks

	Mean Rank
RootedDAGMultiplePath	2.85
AllLevelDAGMultiplePath	2.85
TwoLevelDAGMultiplePath	2.50
MaxLevelDAGMultiplePath	3.50
MinLevelDAGMultiplePath	3.30

Test Statistics[a]

N	10
Chi-Square	3.848
df	4
Asymp. Sig.	.427

a. Friedman Test

FIGURE C.24: Results of Friedman test used to compare the different DAG models with respect to the Multiple Path strategy

to each structure (even if the improvement was not considered statistically significant when compared to other variations). In addition these variations were the most efficient with respect to each structure.

With respect to the statistical significance comparison between the two models, Binary Tree and DAG, to conduct a fair comparison we first compared the two models with respect to the Single Path Strategy and then with respect to the Multiple Path strategy. Regarding the comparison of the two models with respect to the Single Path strategy the Wilcoxon test results are presented in Figure C.25. From the figure it can be seen that, there was a statistically significant performance difference between the two classification models, the DAG classification model was significantly more effective than the Binary Tree hierarchical classification model; z = -2.848 and p = 0.004 ($p < 0.05$).

In context of comparing the two models with respect to the Multiple Path strategy Figure C.26 presents the results obtained from a Wilcoxon test. From the figure it can be seen that, as before, there was a significant difference in performance between the two models, the DAG model was again found to be significantly more effective than the Binary Tree model; z = -1.994 and p = 0.046 ($p < 0.05$).

From the above it can be thus concluded that the DAG model is a better hierarchical classification model when compared to the Binary Tree model, regardless of whether the Single or Multiple Path strategy is adopted. The suggested reason for this is that the DAG model provides for greater flexibility than in the case of the binary tree model, because of the overlap between class groups represented by nodes at the same level in the hierarchy. The consequence of this is that the overlap partly mitigates against the early mis-classification issue. In addition, pruning the weak classifiers from the DAG model

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
DAGSinglePath - BinaryTreeSinglePath	Negative	1[a]	1.00	1.00
	Positive	10[b]	6.50	65.00
	Ties	1[c]		
	Total	12		

a. DAGSinglePath < BinaryTreeSinglePath
b. DAGSinglePath > BinaryTreeSinglePath
c. DAGSinglePath = BinaryTreeSinglePath

Test Statistics[a]

	DAGSinglePath - BinaryTreeSinglePath
Z	-2.848[b]
Asymp. Sig. (2-tailed)	.004

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

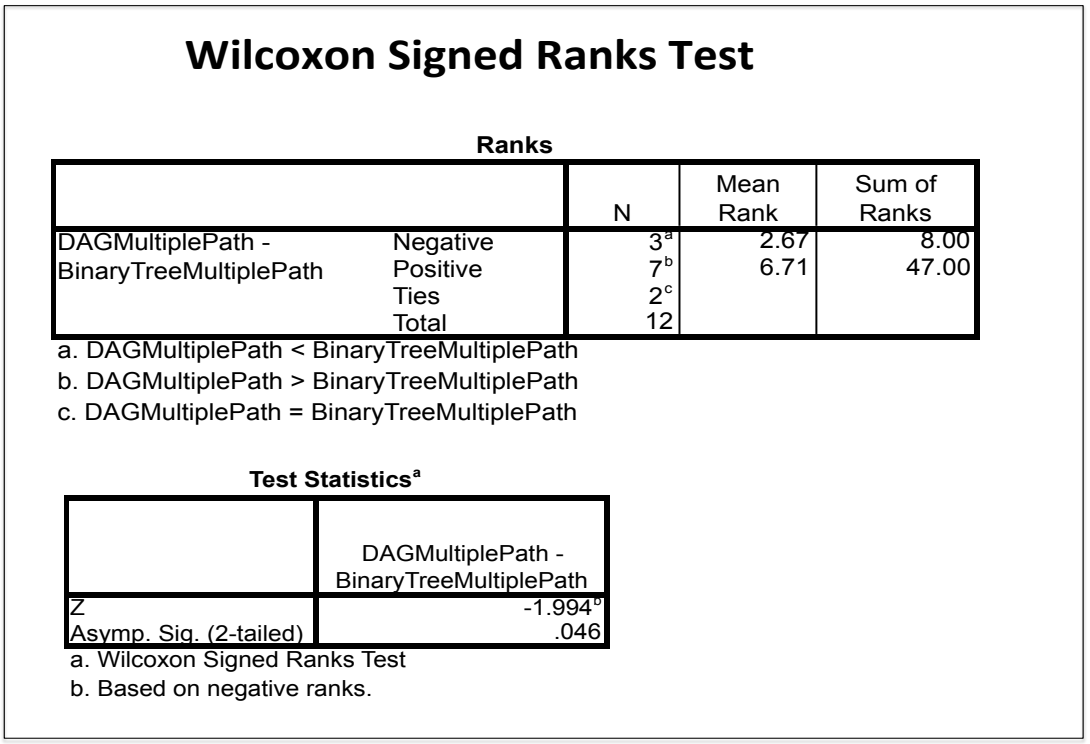
FIGURE C.25: Results of the Wilcoxon test used to compare the Binary Tree hierarchical classification Model and the DAG classification model with respect to the Single Path strategy

results in a better classification accuracy than in the case of the binary tree structure where all the classifiers were used.

C.5 Comparison Between the Hierarchical Classification Model and Conventional Ensemble Classification Models

In this section a comparison between the Min-level DAG classification model, shown in the foregoing to be more effective than the Binary Tree model, and a number of conventional classification models is presented. As before Naive Bayes classification was used for the comparison. Consequently stand alone Naive Bayes classification and Bagging of Naive Bayes classifiers were the conventional classification models adopted. In addition, a "non-consistent" comparison between the Naive Bayes based Min-level DAG model and OVO SVM was conducted. The objective of this last comparison was to compare the suggested model with one of the state of the art methods for multi-class classification. For the comparison the Multiple Path strategy was used through out (see previous discussion presented in Section C.3.3).

Commencing with the comparison between the Naive Bayes based DAG, stand-alone Naive Bayes classification, and Bagging of Naive Bayes classifiers the Friedman test



Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
DAGMultiplePath - BinaryTreeMultiplePath	Negative	3[a]	2.67	8.00
	Positive	7[b]	6.71	47.00
	Ties	2[c]		
	Total	12		

a. DAGMultiplePath < BinaryTreeMultiplePath
b. DAGMultiplePath > BinaryTreeMultiplePath
c. DAGMultiplePath = BinaryTreeMultiplePath

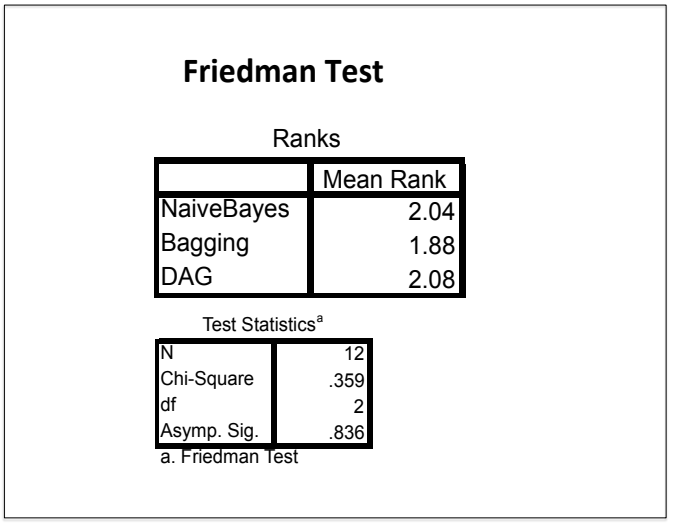
Test Statistics[a]

	DAGMultiplePath - BinaryTreeMultiplePath
Z	-1.994[b]
Asymp. Sig. (2-tailed)	.046

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.26: Results of the Wilcoxon test used to compare the Binary Tree hierarchical classification Model and the DAG classification model with respect to the Multiple Path strategy

results are presented in Figure C.27. The results demonstrate that there was no statistically significant difference between the three considered classifiers ($X^2(2) = 0.359$, $p = 0.836$); although the DAG classification model improved the classification effectiveness for some of the considered datasets.



Friedman Test

Ranks

	Mean Rank
NaiveBayes	2.04
Bagging	1.88
DAG	2.08

Test Statistics[a]

N	12
Chi-Square	.359
df	2
Asymp. Sig.	.836

a. Friedman Test

FIGURE C.27: Results of Friedman test used to compare the DAG classification model, stand-alone Naive Bayes classification and Bagging classification

Regarding the comparison between the DAG classification model and the OVO SVM Figure C.28 presents the Wilcoxon test results. According to Wilcoxon signed rank test, there was again no statistically significant difference in performance between the two classifiers; z = -0.847 and p = 0.397.

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
OVOSVM - NaiveBayesDAG	Negative	5[a]	4.70	23.50
	Positive	6[b]	7.08	42.50
	Ties	1[c]		
	Total	12		

a. OVOSVM < NaiveBayesDAG
b. OVOSVM > NaiveBayesDAG
c. OVOSVM = NaiveBayesDAG

Test Statistics[a]

	OVOSVM - NaiveBayesDAG
Z	-.847[b]
Asymp. Sig. (2-tailed)	.397

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

FIGURE C.28: Results of Wilcoxon test used to compare the DAG classification Model and OVO SVM

From the above we can conclude that the DAG classification model does not statistically outperform the conventional methods for multi-class classification. However, it has a comparable classification effectiveness compared to Naive Bayes, Bagging, and OVO SVM.

C.6 Summary

A precise and comprehensive statistical comparison between the different hierarchical ensemble classification structures, strategies, techniques, and mechanisms considered in this thesis has been presented in this appendix. The Wilcoxon signed ranks test was used for comparing two classifiers; while the Friedman test, with a Nemenyi post-hoc test, was used for comparing several classifiers (more than two). The conducted statistical comparisons were organised as follows:

Binary Tree hierarchical classification model comparisons. The comparisons included comparing: (i) the usage of different classifier (Decision tree, Naive Bayes, and CARM) at the tree nodes, (ii) the different data distribution techniques (k-means clustering, *divisive* hierarchical clustering and data splitting), (iii) the two

classification strategies (Single and Multiple Path), and (iv) the two alternative mechanisms for arriving at a final classification decision with respect to the Multiple Path strategy (BIP and NAP).

DAG classification model comparisons. The comparisons included comparing: (i) the usage of different classifiers (Decision tree, Naive Bayes, and CARM) at the DAG nodes, (ii) the different DAG approaches (Rooted DAG, All-level DAG, Two-level DAG, Max-level DAG, and Min-level DAG), (iii) the two classification strategies (Single and Multiple Path) and (iv) the two alternative mechanisms for arriving at a final classification decision with respect to the Multiple Path strategy (Voting and NAP).

DAG based hierarchical classification versus Binary Tree based hierarchical classification. Statistical comparison of the two models with respect to both the Single and Multiple Path strategies.

Comparison with more conventional models. Comparison of the proposed DAG Hierarchical classification model with more conventional ensemble classification models and stand alone classification.

As a consequence of the evaluation, and according to the reported statistical test results, it was demonstrated that:

1. The most effective classifier with which to generate the classifiers held at the nodes in the case of the Binary Tree hierarchical classification model was found to be Naive Bayes classification (regardless of whether a Single or Multiple Path strategy was adopted).

2. Regarding the comparison between the data distribution techniques to generate the Binary Tree hierarchical classification model, it was found that data splitting and k-means were more effective than hierarchical clustering regardless of the classification strategy adopted (Single or Multiple Path).

3. With respect to the three considered mechanisms for arriving at a final classification decision in context of the Multiple Path strategies and the Binary Tree hierarchical classification model it was demonstrated that the NAP mechanism significantly outperformed the Voting mechanism. No significant difference was detected between the BIP and the NAP mechanisms. Thus, assigning *weight* to the "candidate classes" tends to be more effective than voting.

4. Following multiple paths within the Binary Tree classification model was significantly more effective than following only a single path. In addition it was demonstrated that the Multiple Path strategy could successfully address the mis-classification issue (the number of mis-classification was clearly higher when using a Single Path strategy).

5. The most effective classifier, to generate the DAG classification model, was found to be Naive Bayes classification regardless of the adopted classification strategy (Single or Multiple Path) or even the adopted DAG approach.

6. Regarding the comparison between the different mechanisms for arriving at a final classification decision (Voting, BIP and NAP) when following multiple paths within the DAG, it was found that there was no significant difference in performance between the three mechanisms.

7. Unlike in the case of the Binary Tree hierarchical classification model, following multiple paths within the DAG classification model was found to be not significantly more effective than when following only a single path. The reason for this was argued to be that the combination techniques used to distribute classes between nodes resulted in well-defined class labels at each DAG node, unlike the clustering algorithms that were used with respect to the Binary Tree model; consequently the mis-classification was less and the effect of following multiple paths within the DAG was not highly significant.

8. Although there was a differences in the effectiveness among the considered DAG approaches, these differences were not found to be statistically significant, regardless of the adopted classification strategy (Single or Multiple Path) according to the Friedman statistical tests conducted.

9. According to the conducted statistical tests, usage of the DAG structure was found to be more effective with respect to the generation of the hierarchical classification model than the Binary Tree structure, regardless of the adopted classification strategy (Single or Multiple Path).

10. With respect to the comparison between Stand-alone classification, Bagging and DAG classification, although the DAG classification model improved the classification effectiveness for some of the considered datasets, the obtained improvement with respect to all of the considered datasets was not found to be statistically significant according to the Friedman tests conducted.

11. Regarding the comparison between the DAG classification model and OVO SVM, the reported results from statistical test demonstrated that there was no statistically significant difference between them. However, and as reported in Chapter 6, it seems that OVO SVM is more effective than the DAG classification model for some datasets that feature large numbers of class labels such as Chess KRvK and Letter Recognition.

12. Finally, we can conclude that the DAG classification model does not significantly outperform the conventional methods for multi-class classification. However, it has a comparable classification effectiveness with respect to these well-established

conventional methods such as stand alone Naive Bayes classification, Bagging and OVO SVM.

The statistical tests reported on in this chapter demonstrated that there was no statistically significant difference in operation between the DAG classification model and OVO SVM (as noted above OVO SVM outperformed the DAG model with respect to the Chess KRvK and Letter Recognition data sets). It was therefore suggested that a combination of both models (OVO SVM and DAG) might be appropriate. The expectation is that such a model will improve the classification effectiveness. However, because of the processing power required, this OVO SVM based DAG model can only be realised if it is implemented using some form of distributed or parallel computing. This possibility is thus considered in further detail in the next chapter.

Appendix D

Determining the Best Threshold Value (σ) for Following Multiple Paths within the Binary Tree Hierarchical Classification Model

In this appendix the results produced when using the Multiple Path strategy, in context of Binary Tree hierarchies, with respect to the three considered class label selection mechanisms: (i) BIP (BIC), (ii) NAP (NAC), and (iii) Voting, and the three considered data distribution techniques: (i) k-means, (ii) data splitting, and (iii) *divisive* hierarchical clustering, using a range of values for σ are presented.

TABLE D.1: Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to k-mean data distribution technique, using a range of values for σ

Data set	σ											
	0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	68.42	0.69	68.42	0.69	68.38	0.68	67.80	0.68	66.80	0.67	44.3	0.45
Wine	92.54	0.93	92.54	0.93	92.54	0.93	92.54	0.93	91.36	0.92	89.10	0.89
Nursery	91.9	0.56	91.86	0.56	91.07	0.57	89.71	0.57	88.21	0.56	48.01	0.35
Heart	53.42	0.37	53.01	0.38	45.70	0.34	41.97	0.32	35.21	0.29	33.62	0.25
PageBlocks	92.34	0.44	92.02	0.44	89.75	0.42	89.75	0.42	89.71	0.44	89.38	0.45
Dermatology	77.50	0.76	72.92	0.71	67.60	0.65	63.65	0.61	55.87	0.56	28.66	0.26
Glass	59.73	0.40	55.52	0.38	50.91	0.36	42.02	0.31	43.30	0.31	39.09	0.27
Zoo	95.09	0.60	95.09	0.60	95.09	0.60	96.09	0.61	95.09	0.60	90.27	0.58
Ecoli	76.75	0.32	72.14	0.30	72.05	0.30	72.45	0.30	73.66	0.33	73.66	0.33
Led	52.69	0.53	59.66	0.59	49.06	0.49	46.31	0.47	45.59	0.46	45.25	0.46
PenDigits	82.66	0.82	82.46	0.82	81.67	0.81	79.34	0.79	75.19	0.75	48.67	0.49
Soybean	79.00	0.84	78.82	0.84	77.03	0.83	75.44	0.83	74.36	0.82	49.1	0.51
ChessKRvK	45.26	0.42	41.72	0.38	33.56	0.31	25.77	0.23	23.47	0.20	21.74	0.17
LetterRecog	41.32	0.41	41.06	0.41	40.37	0.40	39.35	0.39	37.59	0.38	20.26	0.20
Mean	72.04	**0.58**	71.23	0.57	68.20	0.55	65.87	0.53	63.96	0.52	51.51	0.40

TABLE D.2: Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to data splitting technique, using a range of values for σ

Data set	σ																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.1×10^{-7}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.98	0.75	74.98	0.75	74.98	0.75
Wine	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	95.67	0.96	95.67	0.96	95.67	0.96
Nursery	90.12	0.44	90.12	0.44	90.12	0.44	90.12	0.44	90.12	0.44	87.41	0.58	87.41	0.58	87.41	0.58	87.41	0.58
Heart	57.70	0.41	57.70	0.41	53.08	0.38	53.08	0.38	53.08	0.38	53.08	0.38	53.08	0.38	51.19	0.37	51.14	0.36
PageBlocks	91.96	0.34	91.96	0.34	91.96	0.34	91.30	0.47	90.10	0.45	90.10	0.45	90.10	0.45	89.50	0.43	89.50	0.43
Dermatology	79.80	0.79	79.80	0.79	79.80	0.79	79.80	0.79	82.94	0.82	82.94	0.82	82.94	0.82	83.46	0.83	84.26	0.84
Glass	63.94	0.43	63.94	0.43	59.73	0.47	57.90	0.48	57.90	0.48	50.99	0.49	50.99	0.49	50.99	0.49	50.99	0.49
Zoo	93.18	0.59	93.18	0.59	93.18	0.59	92.18	0.58	92.18	0.58	92.18	0.58	91.27	0.58	90.32	0.57	90.32	0.57
Ecoli	82.31	0.36	82.31	0.36	80.55	0.35	80.55	0.35	68.65	0.34	68.65	0.34	68.65	0.34	64.15	0.27	60.38	0.20
Led	60.16	0.60	60.16	0.60	60.41	0.60	60.41	0.60	60.41	0.60	60.41	0.60	60.41	0.60	60.41	0.60	60.41	0.60
PenDigits	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	83.30	0.83	83.41	0.83	83.41	0.83
Soybean	79.55	0.81	79.55	0.81	79.55	0.81	79.55	0.81	75.45	0.73	75.45	0.73	75.45	0.73	75.45	0.73	75.45	0.73
chessKRvK	35.18	0.27	35.18	0.27	35.18	0.27	35.18	0.27	28.58	0.35	28.58	0.35	28.58	0.35	30.18	0.35	30.18	0.35
LetterRecog	39.16	0.39	39.16	0.39	39.16	0.39	39.16	0.39	39.16	0.39	43.59	0.43	54.59	0.54	54.59	0.54	54.59	0.54
Mean	72.17	0.56	72.17	0.56	71.43	0.56	71.18	0.57	69.71	0.57	69.34	0.58	71.24	**0.60**	70.84	0.59	70.62	0.59

TABLE D.3: Accuracy and AUC values produced when using the Multiple Path strategy and the best individual probability class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ

Data set	σ															
	0.1×10^{-1}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-6}		0.1×10^{-8}		0.1×10^{-10}		0.1×10^{-15}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	53.36	0.53
Wine	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	30.31	0.36	54.82	0.57
Nursery	31.80	0.17	31.80	0.17	31.80	0.17	31.80	0.17	31.80	0.17	36.40	0.19	36.29	0.19	46.49	0.30
Heart	19.39	0.20	19.39	0.20	18.70	0.21	19.39	0.21	21.46	0.24	21.46	0.24	21.46	0.23	23.39	0.22
PageBlocks	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.96	0.24	2.03	0.28	82.38	0.34
Dermatology	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	41.95	0.33
Glass	15.01	0.12	15.01	0.12	15.01	0.12	18.34	0.14	18.34	0.14	34.48	0.19	38.29	0.22	48.06	0.34
Zoo	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	80.36	0.51
Ecoli	15.17	0.13	15.17	0.13	15.17	0.13	15.17	0.13	15.17	0.13	19.11	0.18	19.98	0.20	43.68	0.27
Led	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	17.41	0.18	44.72	0.44
PenDigits	10.69	0.10	10.69	0.10	10.69	0.10	10.69	0.10	11.95	0.12	11.95	0.12	11.95	0.12	59.58	0.60
Soybean	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	60.67	0.69
ChessKRvK	5.69	0.06	5.69	0.06	5.69	0.06	5.69	0.06	5.69	0.06	4.03	0.06	2.87	0.08	19.92	0.22
LetterRecog	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	28.89	0.29
Mean	15.77	0.17	15.77	0.17	15.72	0.17	16.01	0.17	16.25	0.17	17.91	0.18	18.46	0.19	**49.16**	**0.40**

TABLE D.4: Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to k-mean data distribution technique, using a range of values for σ

Data set	σ											
	0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	68.42	0.69	68.42	0.69	68.40	0.68	67.92	0.68	66.92	0.67	44.14	0.44
Wine	92.54	0.93	92.54	0.93	92.54	0.93	92.54	0.93	91.36	0.92	89.01	0.89
Nursery	91.90	0.56	91.88	0.57	91.10	0.57	90.16	0.57	89.28	0.56	63.39	0.42
Heart	53.07	0.37	53.01	0.38	47.42	0.35	44.39	0.34	37.56	0.30	36.03	0.26
PageBlocks	92.34	0.44	92.09	0.45	89.86	0.43	89.86	0.43	89.82	0.44	89.84	0.46
Dermatology	77.73	0.75	74.53	0.71	69.73	0.66	65.78	0.63	57.42	0.57	33.31	0.31
Glass	59.73	0.40	57.35	0.39	54.10	0.38	46.23	0.33	46.55	0.33	43.30	0.29
Zoo	95.09	0.60	95.09	0.60	95.09	0.60	96.09	0.61	95.09	0.61	90.27	0.58
Ecoli	76.99	0.32	74.62	0.32	75.43	0.32	74.57	0.32	75.17	0.34	75.17	0.34
Led	60.22	0.60	70.72	0.71	69.44	0.69	67.34	0.67	67.19	0.67	66.81	0.67
PenDigits	82.66	0.82	82.47	0.82	81.85	0.82	79.97	0.80	76.78	0.77	52.97	0.53
Soybean	79.00	0.84	78.82	0.84	76.67	0.83	75.08	0.82	74.36	0.82	53.00	0.55
chessKRvK	45.57	0.42	42.79	0.39	36.23	0.34	28.89	0.27	25.45	0.25	22.99	0.23
LetterRecog	41.40	0.41	41.20	0.41	40.76	0.41	40.14	0.40	39.58	0.40	24.84	0.25
Mean	72.62	0.58	72.54	**0.59**	70.62	0.57	68.50	0.56	66.61	0.55	56.08	0.44

TABLE D.5: Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to data splitting technique, using a range of values for σ

Data set	σ																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.1×10^{-7}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	76.44	0.76	76.44	0.76	76.44	0.76
Wine	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	95.08	0.95	96.26	0.96	96.26	0.96	96.26	0.96
Nursery	90.12	0.44	90.12	0.44	90.12	0.44	90.12	0.44	89.09	0.58	89.09	0.58	89.09	0.58	89.09	0.58	89.09	0.58
Heart	57.70	0.41	57.70	0.41	55.57	0.38	55.50	0.38	54.12	0.37	54.12	0.37	53.77	0.36	50.12	0.30	49.87	0.30
PageBlocks	91.96	0.34	91.96	0.34	92.53	0.45	91.76	0.46	91.43	0.45	91.30	0.47	91.27	0.48	91.76	0.45	91.76	0.45
Dermatology	79.80	0.79	79.80	0.79	79.80	0.79	83.46	0.83	85.18	0.85	84.60	0.84	84.60	0.84	84.60	0.84	84.60	0.84
Glass	63.94	0.43	63.94	0.43	62.11	0.45	59.73	0.47	57.90	0.48	57.10	0.50	55.28	0.51	55.28	0.51	55.28	0.51
Zoo	93.18	0.59	93.18	0.59	93.18	0.59	92.18	0.58	92.18	0.58	92.18	0.58	92.18	0.58	92.18	0.58	90.14	0.56
Ecoli	82.31	0.36	82.00	0.36	78.76	0.36	76.39	0.35	67.23	0.29	64.15	0.27	64.15	0.27	60.38	0.20	58.38	0.20
Led	60.16	0.60	60.16	0.60	61.28	0.61	61.13	0.61	61.13	0.61	61.13	0.61	61.13	0.61	73.13	0.73	73.13	0.73
PenDigits	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	81.18	0.81	81.18	0.81	81.18	0.81
Soybean	79.55	0.81	79.55	0.81	79.55	0.81	79.55	0.81	83.44	0.82	83.44	0.82	83.71	0.83	83.71	0.83	85.62	0.84
ChessKRvK	35.18	0.27	35.18	0.27	35.18	0.27	35.16	0.27	36.68	0.35	33.93	0.37	33.88	0.37	33.88	0.37	33.88	0.37
LetterRecog	39.16	0.39	39.16	0.39	39.16	0.39	39.18	0.39	39.44	0.39	41.61	0.42	53.44	0.53	53.44	0.53	53.44	0.53
Mean	72.17	0.56	72.15	0.56	71.76	0.57	71.54	0.57	71.08	0.58	70.76	0.59	72.60	**0.61**	72.96	0.60	72.79	0.60

TABLE D.6: Accuracy and AUC values produced when using the Multiple Path strategy and the best normalised accumulated probability class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ

Data set	σ															
	0.1×10^{-1}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-6}		0.1×10^{-8}		0.1×10^{-10}		0.1×10^{-15}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	53.62	0.53
Wine	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	30.31	0.36	56.59	0.58
Nursery	31.80	0.17	31.80	0.17	31.80	0.17	31.80	0.17	31.80	0.17	36.41	0.19	36.29	0.19	46.37	0.30
Heart	19.39	0.20	19.39	0.20	18.70	0.21	20.80	0.22	20.80	0.22	22.10	0.25	22.10	0.25	24.43	0.22
PageBlocks	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.96	0.24	2.03	0.28	82.38	0.36
Dermatology	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	43.28	0.34
Glass	15.01	0.12	15.01	0.12	15.01	0.12	18.34	0.14	18.34	0.14	34.48	0.19	39.71	0.23	46.23	0.34
Zoo	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	84.18	0.52
Ecoli	15.17	0.13	15.17	0.13	15.17	0.13	15.17	0.13	15.17	0.13	17.90	0.17	21.79	0.22	44.59	0.28
Led	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	17.59	0.18	44.16	0.44
PenDigits	10.69	0.10	10.69	0.10	10.69	0.10	10.69	0.10	7.30	0.07	12.22	0.12	12.22	0.12	60.80	0.61
Soybean	7.30	0.07	7.30	0.07	7.30	0.07	4.26	0.06	4.26	0.06	7.30	0.07	7.30	0.07	64.23	0.71
chessKRvK	5.69	0.06	5.69	0.06	5.69	0.06	5.69	0.06	5.69	0.06	4.33	0.06	3.09	0.08	18.59	0.21
LetterRecog	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	3.09	0.08	4.85	0.05	30.41	0.30
Mean	15.77	0.17	15.77	0.17	15.72	0.17	15.89	0.17	15.65	0.17	17.78	0.19	18.79	0.20	**49.99**	**0.41**

TABLE D.7: Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to k-mean data distribution technique, using a range of values for σ

Data set	σ											
	0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	68.42	0.69	68.42	0.69	68.46	0.69	68.34	0.68	68.12	0.68	46.78	0.47
Wine	92.54	0.93	92.54	0.93	92.54	0.93	92.54	0.93	92.54	0.93	68.96	0.64
Nursery	91.91	0.56	91.57	0.56	89.02	0.55	85.85	0.53	85.52	0.49	33.79	0.21
Heart	53.49	0.36	54.52	0.39	52.45	0.35	45.07	0.32	28.71	0.25	15.53	0.18
PageBlocks	91.56	0.31	73.70	0.24	13.93	0.28	13.71	0.29	13.69	0.23	13.62	0.25
Dermatology	79.62	0.78	75.67	0.73	69.60	0.66	67.89	0.65	64.46	0.64	22.38	0.24
Glass	62.51	0.41	57.83	0.37	48.76	0.28	43.12	0.25	42.97	0.22	42.82	0.15
Zoo	95.09	0.60	95.09	0.60	95.09	0.60	93.09	0.58	85.09	0.49	13.91	0.09
Ecoli	78.07	0.31	71.24	0.28	75.62	0.29	58.34	0.24	58.65	0.23	55.36	0.20
Led	48.13	0.48	49.28	0.49	32.47	0.32	34.63	0.35	30.22	0.31	24.34	0.25
PenDigits	82.66	0.82	82.53	0.82	82.44	0.82	81.38	0.81	79.08	0.79	12.19	0.12
Soybean	79.00	0.84	79.00	0.84	79.34	0.84	79.16	0.84	80.58	0.85	20.14	0.09
chessKRvK	45.11	0.41	40.57	0.36	34.54	0.29	30.14	0.22	27.20	0.15	15.92	0.06
LetterRecog	41.30	0.41	40.84	0.41	39.80	0.40	38.30	0.38	36.36	0.36	6.10	0.06
Mean	**72.10**	**0.57**	69.49	0.55	62.43	0.52	59.40	0.51	56.66	0.47	27.99	0.22

TABLE D.8: Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to data splitting technique, using a range of values for σ

Data set	σ													
	0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.1×10^{-7}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.30	0.74	74.28	0.74	56.80	0.57
Wine	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	94.49	0.95	70.92	0.65
Nursery	90.12	0.44	90.12	0.44	90.12	0.44	90.12	0.44	65.85	0.47	63.33	0.35	33.35	0.20
Heart	57.70	0.41	57.36	0.40	56.80	0.39	59.63	0.38	60.25	0.38	58.18	0.34	57.42	0.29
PageBlocks	91.96	0.34	91.96	0.34	91.98	0.32	92.00	0.32	92.05	0.34	91.98	0.32	91.85	0.30
Dermatology	79.80	0.79	79.51	0.79	72.41	0.69	64.90	0.57	55.01	0.47	53.87	0.44	50.54	0.32
Glass	63.94	0.43	63.54	0.42	65.05	0.42	64.57	0.39	60.84	0.31	57.43	0.23	56.48	0.19
Zoo	93.18	0.59	92.18	0.58	88.18	0.54	82.18	0.46	81.09	0.43	81.09	0.43	60.18	0.23
Ecoli	82.31	0.36	72.19	0.31	65.92	0.24	63.85	0.19	64.80	0.19	64.20	0.17	64.20	0.17
Led	47.38	0.48	30.13	0.29	21.25	0.20	20.59	0.19	20.59	0.19	20.59	0.19	20.59	0.19
PenDigits	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	68.56	0.68	20.63	0.20
Soybean	79.55	0.81	79.55	0.81	79.55	0.81	79.55	0.81	79.55	0.81	80.25	0.81	7.12	0.13
chessKRvK	35.18	0.27	35.18	0.27	35.14	0.27	24.68	0.17	9.39	0.10	9.94	0.10	9.94	0.10
LetterRecog	39.16	0.39	39.16	0.39	39.18	0.39	38.07	0.38	36.41	0.36	32.87	0.33	7.36	0.07
Mean	**71.26**	**0.55**	69.16	0.53	67.35	0.51	65.54	0.48	61.66	0.46	60.79	0.43	43.38	0.26

TABLE D.9: Accuracy and AUC values produced when using the Multiple Path strategy and voting class label mechanism with respect to *divisive* hierarchical clustering technique, using a range of values for σ

Data set	σ													
	0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-6}		0.1×10^{-8}		0.1×10^{-10}		0.1×10^{-15}		0.0	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	43.12	0.43	55.90	0.56
Wine	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	28.31	0.34	31.51	0.36	46.80	0.48
Nursery	31.80	0.17	31.80	0.17	31.80	0.17	34.80	0.17	34.81	0.18	39.48	0.22	42.36	0.28
Heart	19.39	0.20	19.39	0.20	19.73	0.20	20.32	0.19	30.35	0.21	30.35	0.21	11.18	0.17
PageBlocks	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.68	0.22	1.90	0.24	82.05	0.26
Dermatology	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	10.60	0.17	41.22	0.30
Glass	15.01	0.12	15.01	0.12	15.01	0.12	28.44	0.12	28.44	0.12	41.31	0.13	46.48	0.16
Zoo	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	12.00	0.13	26.00	0.13
Ecoli	15.17	0.13	15.17	0.13	15.17	0.13	20.02	0.14	20.02	0.14	19.98	0.18	55.92	0.23
Led	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.16	0.15	15.63	0.16	25.66	0.26
PenDigits	10.69	0.10	10.69	0.10	10.69	0.10	10.69	0.10	10.69	0.10	10.69	0.10	23.11	0.24
Soybean	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	7.30	0.07	7.03	0.07	24.93	0.14
chessKRvK	5.69	0.06	5.69	0.06	4.74	0.06	4.74	0.06	4.75	0.06	2.97	0.06	14.61	0.06
LetterRecog	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	4.85	0.05	9.02	0.09
Mean	15.77	0.17	15.77	0.17	15.73	0.17	17.29	0.17	18.01	0.17	19.39	0.18	**36.09**	**0.24**

TABLE D.10: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	53.60	0.53	59.22	0.59	59.02	0.59	58.92	0.59	58.92	0.59
Wine	78.99	0.81	78.99	0.81	78.99	0.81	78.99	0.81	78.99	0.81
Nursery	80.05	0.43	82.91	0.43	82.53	0.42	84.41	0.43	84.41	0.43
Heart	46.66	0.22	52.80	0.23	53.76	0.24	53.76	0.24	53.76	0.24
PageBlocks	90.79	0.24	90.79	0.24	90.79	0.24	90.79	0.24	90.79	0.24
Dermatology	70.12	0.60	71.96	0.62	75.73	0.65	74.92	0.64	74.92	0.64
Glass	46.23	0.30	48.93	0.29	48.46	0.29	48.46	0.29	48.46	0.29
Zoo	87.00	0.51	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52
Ecoli	63.29	0.28	64.55	0.28	64.55	0.28	65.15	0.28	65.15	0.28
Led	24.41	0.23	43.69	0.43	52.16	0.52	54.78	0.55	54.78	0.55
PenDigits	54.95	0.54	60.65	0.61	61.12	0.61	61.12	0.61	61.12	0.61
Soybean	75.29	0.73	78.13	0.75	79.01	0.76	79.01	0.76	79.01	0.76
ChessKRvK	20.82	0.10	28.15	0.15	32.16	0.17	32.99	0.18	32.99	0.18
LetterRecog	15.71	0.16	24.74	0.25	29.17	0.29	29.58	0.30	29.58	0.30
Mean	57.71	0.41	62.39	0.44	63.96	0.46	64.35	0.46	64.35	0.46

TABLE D.11: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	68.84	0.68	63.18	0.64	57.12	0.58	57.02	0.58	57.02	0.58
Wine	77.39	0.74	78.59	0.76	78.59	0.76	78.59	0.76	78.59	0.76
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43
Heart	53.07	0.21	52.39	0.20	52.39	0.20	52.39	0.20	52.39	0.20
PageBlocks	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20
Dermatology	75.00	0.64	67.55	0.55	60.28	0.46	60.28	0.46	60.28	0.46
Glass	61.96	0.34	61.56	0.31	61.56	0.31	61.56	0.31	61.56	0.31
Zoo	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	65.52	0.18	67.86	0.20	66.57	0.20	66.57	0.20	66.57	0.20
Led	23.91	0.23	29.19	0.28	41.38	0.41	45.53	0.45	45.53	0.45
PenDigits	37.37	0.37	42.44	0.42	43.89	0.43	42.68	0.42	42.68	0.42
Soybean	75.10	0.81	84.88	0.87	88.43	0.89	88.97	0.89	88.97	0.89
chessKRvK	10.18	0.06	7.08	0.06	26.58	0.13	28.13	0.14	28.13	0.14
LetterRecog	17.48	0.18	23.43	0.24	29.13	0.29	33.12	0.32	33.12	0.32
Mean	59.10	0.40	59.98	0.40	61.96	0.41	62.60	0.42	62.60	0.42

TABLE D.12: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best individual confidence class label selection mechanism, with respect to a CARM and *divisive* hierarchical clustering generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	50.68	0.51	61.04	0.61	61.58	0.61	61.62	0.61	61.62	0.61
Wine	83.29	0.85	84.28	0.86	84.28	0.86	84.28	0.86	84.28	0.86
Nursery	51.97	0.27	53.67	0.27	53.12	0.27	53.07	0.27	53.07	0.27
Heart	51.63	0.22	53.63	0.22	53.35	0.23	53.35	0.23	53.35	0.23
PageBlocks	74.15	0.24	73.88	0.23	73.88	0.23	73.73	0.25	73.73	0.25
Dermatology	57.35	0.49	68.87	0.63	71.16	0.66	71.39	0.66	71.39	0.66
Glass	44.80	0.23	47.18	0.27	51.24	0.30	52.19	0.30	52.19	0.30
Zoo	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	58.67	0.23	57.72	0.23	51.86	0.20	52.17	0.21	52.17	0.21
Led	21.69	0.22	29.56	0.29	35.38	0.35	35.16	0.35	35.16	0.35
PenDigits	35.86	0.36	46.76	0.47	50.41	0.50	50.74	0.51	50.74	0.51
Soybean	57.18	0.52	58.60	0.53	56.82	0.52	57.36	0.52	57.36	0.52
chessKRvK	11.22	0.11	22.26	0.13	22.48	0.11	22.50	0.11	22.50	0.11
LetterRecog	18.46	0.18	20.51	0.21	21.67	0.22	21.54	0.22	21.54	0.22
Mean	50.14	0.35	54.50	0.39	55.16	0.40	55.29	0.40	55.29	0.40

TABLE D.13: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	59.74	0.59	59.06	0.59	59.02	0.59	58.92	0.59	58.92	0.59
Wine	78.99	0.81	78.99	0.81	78.99	0.81	78.99	0.81	78.99	0.81
Nursery	80.32	0.43	82.67	0.44	80.28	0.40	82.99	0.42	84.41	0.43
Heart	47.01	0.30	54.11	0.25	53.76	0.24	53.76	0.24	53.76	0.24
PageBlocks	90.29	0.0.24	90.79	0.24	91.17	0.35	90.79	0.24	90.79	0.24
Dermatology	63.08	0.61	77.85	0.69	77.34	0.68	74.92	0.64	74.92	0.64
Glass	43.85	0.27	47.98	0.29	48.93	0.29	48.46	0.29	48.46	0.29
Zoo	87.00	0.51	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52
Ecoli	64.24	0.29	64.85	0.28	64.85	0.28	64.55	0.28	65.15	0.28
Led	34.72	0.34	56.69	0.56	52.13	0.52	54.78	0.55	54.78	0.55
PenDigits	58.73	0.59	60.60	0.61	60.86	0.61	60.87	0.61	61.12	0.61
Soybean	76.55	0.75	79.01	0.75	79.01	0.76	79.01	0.76	79.01	0.76
chessKRvK	21.51	0.13	28.16	0.16	31.06	0.17	33.02	0.18	32.99	0.18
LetterRecog	10.08	0.10	18.70	0.19	27.87	0.28	29.43	0.29	29.58	0.30
Mean	58.29	0.44	63.39	0.46	63.81	0.46	64.18	0.46	64.35	0.46

TABLE D.14: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	57.44	0.58	57.44	0.58	57.44	0.58	57.02	0.58	57.02	0.58
Wine	78.59	0.76	78.59	0.76	78.59	0.76	78.59	0.76	78.59	0.76
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43
Heart	52.39	0.20	52.39	0.20	52.39	0.20	52.39	0.20	52.39	0.20
PageBlocks	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20
Dermatology	60.28	0.46	60.28	0.46	60.28	0.46	60.28	0.46	60.28	0.46
Glass	60.28	0.46	61.56	0.31	61.56	0.31	61.56	0.31	61.56	0.31
Zoo	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	66.57	0.20	66.57	0.20	66.57	0.20	66.57	0.20	66.57	0.20
Led	32.16	0.32	32.16	0.32	41.53	0.41	45.53	0.45	45.53	0.45
PenDigits	32.22	0.33	40.99	0.41	40.12	0.40	42.68	0.42	42.68	0.42
Soybean	88.97	0.89	88.97	0.89	88.43	0.89	88.97	0.89	88.97	0.89
chessKRvK	14.03	0.09	23.61	0.13	27.28	0.13	28.13	0.14	28.13	0.14
LetterRecog	33.12	0.33	33.12	0.33	27.39	0.27	33.12	0.33	33.12	0.33
Mean	59.83	0.41	61.23	0.41	61.65	0.41	62.60	0.42	62.60	0.42

TABLE D.15: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the best normalised accumulated confidence class label selection mechanism, with respect to a CARM and *divisive* hierarchical clustering generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	46.00	0.46	61.28	0.61	61.58	0.61	61.62	0.61	61.62	0.61
Wine	79.53	0.81	84.28	0.86	84.28	0.86	84.28	0.86	84.28	0.86
Nursery	45.36	0.26	49.54	0.25	53.06	0.27	53.07	0.27	53.07	0.27
Heart	51.35	0.24	53.63	0.23	53.35	0.23	53.35	0.23	53.35	0.23
PageBlocks	74.01	0.26	73.73	0.26	73.75	0.26	73.75	0.26	73.73	0.25
Dermatology	54.33	0.47	68.74	0.64	71.16	0.66	71.39	0.66	71.39	0.66
Glass	44.17	0.22	47.18	0.28	51.71	0.30	52.19	0.30	52.19	0.30
Zoo	74.18	0.42	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	56.20	0.21	54.69	0.20	52.17	0.21	52.17	0.21	52.17	0.21
Led	18.94	0.19	25.59	0.25	37.94	0.38	34.91	0.35	35.16	0.35
PenDigits	32.24	0.33	46.40	0.47	50.43	0.50	50.30	0.51	50.74	0.51
Soybean	43.10	0.45	56.47	0.52	56.83	0.52	57.36	0.52	57.36	0.52
chessKRvK	10.00	0.10	20.48	0.11	22.20	0.11	22.50	0.11	22.50	0.11
LetterRecog	13.04	0.13	18.63	0.19	21.57	0.22	21.54	0.22	21.54	0.22
Mean	45.89	0.33	53.26	0.38	55.36	0.40	55.25	0.40	55.29	0.40

TABLE D.16: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and k-mean generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	59.40	0.59	58.94	0.59	58.90	0.59	58.92	0.59	58.92	0.59
Wine	78.59	0.81	78.99	0.81	78.79	0.81	78.99	0.81	78.99	0.81
Nursery	81.30	0.45	82.62	0.43	82.79	0.42	84.41	0.43	84.41	0.43
Heart	53.76	0.24	53.49	0.23	53.76	0.24	53.76	0.24	53.76	0.24
PageBlocks	90.79	0.24	90.79	0.24	90.79	0.24	90.79	0.24	90.79	0.24
Dermatology	66.87	0.56	68.07	0.57	74.64	0.64	74.92	0.64	74.92	0.64
Glass	46.48	0.26	46.48	0.26	46.48	0.26	48.46	0.29	48.46	0.29
Zoo	87.00	0.51	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52
Ecoli	71.54	0.28	70.68	0.29	70.37	0.29	65.15	0.28	65.15	0.28
Led	31.47	0.31	35.91	0.36	48.56	0.49	54.78	0.55	54.78	0.55
PenDigits	68.27	0.68	66.68	0.67	66.42	0.66	61.12	0.61	61.12	0.61
Soybean	78.13	0.76	77.59	0.75	79.01	0.76	79.01	0.76	79.01	0.76
chessKRvK	26.91	0.20	31.14	0.23	32.13	0.17	32.99	0.18	32.99	0.18
LetterRecog	21.07	0.21	21.24	0.21	28.35	0.28	29.58	0.30	29.58	0.30
Mean	61.54	0.44	62.19	0.44	64.21	0.46	64.35	0.46	64.35	0.46

TABLE D.17: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and data splitting generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	57.02	0.58	57.02	0.58	57.02	0.58	57.02	0.58	57.02	0.58
Wine	77.39	0.74	78.59	0.76	78.59	0.76	78.59	0.76	78.59	0.76
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43
Heart	52.39	0.20	52.39	0.20	52.39	0.20	52.39	0.20	52.39	0.20
PageBlocks	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20
Dermatology	59.76	0.46	60.28	0.46	60.28	0.46	60.28	0.46	60.28	0.46
Glass	58.20	0.20	58.20	0.20	58.20	0.20	61.56	0.31	61.56	0.31
Zoo	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	64.20	0.17	66.27	0.19	66.57	0.20	66.57	0.20	66.57	0.20
Led	19.25	0.18	24.47	0.23	37.53	0.37	45.53	0.45	45.53	0.45
PenDigits	21.49	0.21	36.40	0.36	40.45	0.40	42.68	0.42	42.68	0.42
Soybean	75.10	0.81	84.88	0.87	88.43	0.89	88.97	0.89	88.97	0.89
chessKRvK	10.02	0.06	7.32	0.07	24.66	0.13	28.06	0.14	28.13	0.14
LetterRecog	16.61	0.17	23.16	0.23	29.50	0.29	33.12	0.33	33.12	0.33
Mean	55.22	0.35	57.90	0.38	61.09	0.40	62.60	0.42	62.60	0.42

TABLE D.18: Average Accuracy and AUC values produced when using the Multiple Path strategy coupled with the voting class label selection mechanism, with respect to a CARM and hierarchical clustering generated Binary Tree model, using a range of values for σ

Data set	Threshold Value (σ)									
	$\sigma = 90$		$\sigma = 80$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	56.54	0.56	61.54	0.61	61.54	0.61	61.62	0.61	61.62	0.61
Wine	85.46	0.86	84.28	0.86	84.28	0.86	84.28	0.86	84.28	0.86
Nursery	53.06	0.27	53.06	0.27	53.06	0.27	53.07	0.27	53.07	0.27
Heart	54.11	0.21	53.98	0.23	53.35	0.23	53.35	0.23	53.35	0.23
PageBlocks	74.15	0.24	73.88	0.23	73.88	0.23	73.73	0.25	73.73	0.25
Dermatology	63.14	0.53	70.41	0.65	71.16	0.66	71.39	0.66	71.39	0.66
Glass	50.11	0.27	50.11	0.27	50.11	0.27	52.19	0.30	52.19	0.30
Zoo	76.18	0.45	85.00	0.49	85.00	0.49	85.00	0.49	85.00	0.49
Ecoli	51.86	0.20	51.86	0.20	51.86	0.20	52.17	0.21	52.17	0.21
Led	29.44	0.29	30.56	0.30	40.41	0.40	35.16	0.35	35.16	0.35
PenDigits	62.54	0.62	52.26	0.52	51.26	0.51	50.74	0.51	50.74	0.51
Soybean	63.02	0.55	59.50	0.53	56.64	0.52	57.36	0.52	57.36	0.52
chessKRvK	29.55	0.19	26.15	0.15	22.80	0.11	22.50	0.11	22.50	0.11
LetterRecog	29.25	0.29	22.78	0.23	21.73	0.22	21.54	0.22	21.54	0.22
Mean	55.60	0.40	55.38	0.40	55.51	0.40	55.29	0.40	55.29	0.40

Appendix E

Following All Possible Paths Within Rooted DAG

In this appendix the algorithm for following all possible paths, greater than a predefined threshold, within a *rooted* DAG is presented (Algorithm 21).

Algorithm 21 Rooted DAG Multi-Path Classification Following All Possible Paths

INPUT
e = A new unseen example
$Root$ = Start node for the DAG
σ = Path selection threshold
OUTPUT
The predicted class label c for the input example e

GLOBAL VARIABLES
$Path = \{\}$ (Set of identified paths each comprised of: (i) a class label and
(ii) an associated normalised Bayesian probability value)

Start
$accumProb = 0.0$ (Accumulated Bayesian probability start value)
$counter = 0$ (Counter for number of probability values in a followed path)
$dagMultiPathClassify(e, Root, accumProb, counter)$
c = Class label with highest probability value in $Path$
End

function $dagMultiPathClassify(e, Node, accumProb, counter)$
    C = Classification result for e using classifier $Node.G_i$
    P = Bayesian probability values associated with each class group in C
    C_1 = Class group in C associated with highest probability value
    p_1 = Bayesian probability associated with C_1
    C_i = Class group in C
    p_i = Bayesian probability associated with C_i
    if $|C_1| == 1$ **then**
        $normProb = (AccumProb + p_1)/(counter + 1)$
        $Path = Path \cup \langle c, normProb \rangle$ $(c \in C_1)$
    else
        $ChildNode$ = child node representing class group C_1
        $dagMultiPathClassify(e, ChildNode, accumProb + p_1, counter + 1)$
    end if
    for $i = 2$ to $|C|$ **do**
        if $p_i \geq \sigma$ **then**
            if $|C_i| == 1$ **then**
                $normProb = (AccumProb + p_i)/(countert + 1)$
                $Path = Path \cup \langle c, normProb \rangle$ $(c \in C_i)$
            else
                $ChildNode$ = child node representing class group C_i
                $dagMultiPathClassify(e, ChildNode, accumProb + p_i, counter + 1)$
            end if
        end if
    end for
end function

Appendix F

Determining the Best Threshold Value With respect to the Rooted DAG Classification Model

Table F.1 presents the effect of using different values for τ with respect to CARM in order to generate a rooted DAG classification model to generate the rooted DAG classification model.

TABLE F.1: Average Accuracy and AUC values obtained using different values for τ with respect to CARM in order to generate a rooted DAG classification model

Data set	Classes	$\tau = 40$		$\tau = 30$		$\tau = 20$		$\tau = 10$		$\tau = 16$	
		Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
Waveform	3	68.50	0.68	68.54	0.69	68.54	0.69	68.54	0.69	68.54	0.69
Wine	3	86.26	0.86	86.26	0.86	86.26	0.86	86.26	0.86	86.26	0.86
Nursery	5	35.86	0.32	66.25	0.32	86.81	0.43	86.81	0.43	86.81	0.43
Heart	5	53.42	0.24	49.63	0.26	52.60	0.20	53.76	0.20	53.76	0.20
PageBlocks	5	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20
Dermatology	6	60.10	0.41	67.42	0.50	79.62	0.71	79.62	0.71	79.62	0.71
Glass	7	51.71	0.18	51.71	0.18	51.71	0.18	61.71	0.36	61.71	0.36
Zoo	7	63.18	0.26	75.09	0.37	83.09	0.46	88.00	0.52	88.00	0.52
Ecoli	8	39.21	0.17	28.28	0.17	32.12	0.24	32.12	0.24	32.12	0.24
Led	10	26.25	0.25	20.75	0.21	35.25	0.36	40.06	0.40	40.06	0.40
PenDigits	10	28.51	0.27	28.51	0.27	28.51	0.27	46.76	0.47	46.76	0.47
Mean		54.80	0.35	57.47	0.37	63.12	0.42	66.67	0.46	**66.67**	**0.46**

TABLE F.2: Accuracy and AUC values produced when using the two branch strategy and the NAP mechanism with respect to a rooted DAG using a range of values for σ

Data set	σ																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.6×10^{-4}		0.7×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77
Wine	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95
Nursery	90.26	0.45	90.28	0.45	90.28	0.45	90.28	0.45	90.17	0.45	90.28	0.45	90.28	0.45	90.17	0.45	90.28	0.45
Heart	55.91	0.35	56.60	0.34	56.60	0.34	56.60	0.34	55.57	0.35	56.60	0.34	56.60	0.34	55.57	0.35	55.37	0.35
PageBlocks	92.69	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52
Dermatology	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85
Glass	69.81	0.46	70.29	0.46	70.29	0.46	70.29	0.46	71.16	0.50	70.29	0.46	70.29	0.46	71.16	0.50	72.99	0.51
Zoo	92.18	0.58	91.18	0.57	91.18	0.57	91.18	0.57	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	92.18	0.58
Ecoli	84.43	0.41	84.38	0.39	84.38	0.39	84.38	0.39	82.26	0.38	84.38	0.39	84.38	0.39	82.26	0.38	82.56	0.38
Led	75.66	0.76	75.47	0.76	75.47	0.76	75.47	0.76	75.56	0.76	75.47	0.76	75.47	0.76	75.56	0.76	75.56	0.76
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83
Soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92
Mean	82.88	0.65	82.87	0.65	82.87	0.65	82.87	0.65	82.85	0.66	83.04	0.65	83.04	0.65	82.85	0.66	**82.94**	**0.66**

TABLE F.3: Accuracy and AUC values produced when using the two branch strategy and the voting mechanism with respect to a rooted DAG using a range of values for σ

Data set	Threshold Value (σ)																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-5}		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	76.88	0.77	77.00	0.77	77.00	0.77
Wine	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	93.91	0.95	95.08	0.95	95.08	0.95
Nursery	90.26	0.45	90.26	0.45	90.26	0.45	90.28	0.45	90.26	0.45	90.26	0.45	90.26	0.45	90.26	0.45	90.28	0.45
Heart	55.91	0.35	55.91	0.35	55.63	0.35	56.32	0.34	57.63	0.37	57.98	0.36	58.67	0.37	57.63	0.37	56.94	0.34
PageBlocks	92.69	0.52	92.69	0.52	92.69	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52
Dermatology	87.23	0.85	87.23	0.85	87.23	0.85	86.94	0.85	87.51	0.85	87.18	0.85	87.46	0.85	86.94	0.85	86.94	0.85
Glass	69.81	0.46	69.81	0.45	69.81	0.45	70.29	0.46	69.81	0.45	70.29	0.45	70.29	0.46	70.76	0.46	69.81	0.45
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	91.18	0.57	91.18	0.57	91.18	0.57	91.18	0.57	91.18	0.57	91.18	0.57
Ecoli	84.43	0.41	84.69	0.41	84.38	0.39	84.38	0.39	84.38	0.39	84.38	0.39	84.38	0.39	84.38	0.39	84.38	0.39
Led	75.66	0.76	75.44	0.76	75.50	0.76	75.47	0.76	75.47	0.76	75.47	0.76	75.47	0.76	75.47	0.76	75.47	0.76
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83
Soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92
Mean	82.88	0.65	82.89	0.65	82.84	0.65	82.83	0.65	82.94	0.65	**82.98**	**0.65**	82.96	0.65	82.97	0.65	82.84	0.65

TABLE F.4: Accuracy and AUC values produced when using the two branch strategy and the BIP mechanism with respect to a rooted DAG using a range of values for σ

Data set	Threshold Value (σ)															
	0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-5}		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77
Wine	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.67	0.96	95.67	0.96	95.08	0.95
Nursery	90.26	0.45	90.26	0.45	90.25	0.45	90.25	0.45	90.25	0.45	90.25	0.45	90.25	0.45	90.28	0.45
Heart	55.91	0.35	55.91	0.35	56.19	0.35	56.19	0.35	55.84	0.35	55.22	0.35	56.19	0.35	56.19	0.35
PageBlocks	92.69	0.52	92.69	0.52	92.69	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52
Dermatology	87.23	0.85	87.51	0.85	87.23	0.85	86.03	0.85	86.03	0.85	85.75	0.85	86.03	0.85	86.94	0.85
Glass	69.81	0.46	70.29	0.46	72.04	0.49	70.69	0.50	70.21	0.50	68.30	0.49	69.43	0.49	72.11	0.50
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	92.27	0.59	92.27	0.59	92.27	0.59	92.27	0.59	93.18	0.59
Ecoli	84.43	0.41	83.78	0.40	82.37	0.38	82.26	0.38	82.26	0.38	82.26	0.38	82.26	0.38	82.56	0.38
Led	75.66	0.76	75.41	0.75	75.41	0.75	75.41	0.75	75.41	0.75	75.41	0.75	75.41	0.75	75.41	0.75
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83
Soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92
Mean	82.88	0.65	82.87	0.65	82.90	0.65	82.68	0.66	82.61	0.66	82.43	0.66	82.62	0.66	**82.98**	**0.66**

TABLE F.5: Accuracy and AUC values produced when using the three branch strategy with and the normalised accumulated weight mechanism with respect to a rooted DAG using a range of values for σ

Data set	Threshold Value (σ)															
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.02	0.77	77.00	0.77
Wine	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95
Nursery	90.26	0.45	90.26	0.45	90.26	0.45	90.65	0.48	88.94	0.58	87.58	0.57	54.54	0.41	91.40	0.53
Heart	55.91	0.35	55.91	0.35	56.60	0.37	56.60	0.38	54.73	0.37	54.39	0.37	53.76	0.38	55.63	0.38
PageBlocks	92.69	0.52	92.69	0.52	91.89	0.53	91.72	0.53	91.69	0.53	91.69	0.53	91.67	0.53	91.69	0.53
Dermatology	87.23	0.85	87.23	0.85	87.23	0.85	86.66	0.85	86.37	0.84	85.51	0.84	84.66	0.84	86.66	0.85
Glass	69.81	0.46	69.81	0.46	67.03	0.46	64.97	0.50	59.09	0.51	54.40	0.49	51.07	0.47	63.22	0.51
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	93.18	0.59	93.18	0.59	93.18	0.59	92.27	0.59	93.18	0.59
Ecoli	84.43	0.41	84.13	0.41	78.46	0.38	71.84	0.33	68.55	0.33	68.55	0.33	68.55	0.33	70.37	0.33
Led	75.66	0.76	75.50	0.76	75.44	0.76	75.41	0.76	75.41	0.76	75.41	0.76	75.41	0.76	75.41	0.76
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	84.33	0.84	83.58	0.83
Soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.22	0.92	90.75	0.92
Mean	82.88	0.65	82.84	0.65	82.13	0.65	81.45	0.66	80.36	**0.67**	79.76	0.66	76.55	0.65	81.16	0.66

TABLE F.6: Accuracy and AUC values produced when using the all branch strategy with and the normalised accumulated weight mechanism with respect to a rooted DAG using a range of values for σ

Data set	Threshold Value (σ)															
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.00	0.77	77.02	0.77	77.00	0.77
Wine	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95	95.08	0.95
Nursery	90.26	0.45	90.26	0.45	90.26	0.45	90.65	0.48	89.48	0.58	88.80	0.58	52.76	0.40	91.40	0.53
Heart	55.91	0.35	55.91	0.35	56.25	0.36	55.98	0.37	54.94	0.37	54.25	0.36	52.94	0.35	55.29	0.37
PageBlocks	92.69	0.52	92.69	0.52	91.89	0.53	91.89	0.53	91.85	0.53	91.85	0.53	91.83	0.53	91.85	0.53
Dermatology	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	86.37	0.84	85.80	0.84	87.23	0.85
Glass	69.81	0.46	69.81	0.46	67.03	0.45	66.40	0.50	61.87	0.51	58.13	0.49	54.80	0.48	66.00	0.52
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	93.18	0.59	93.18	0.59	93.18	0.59	92.27	0.59	93.18	0.59
Ecoli	84.43	0.41	84.43	0.41	79.93	0.38	74.52	0.34	72.45	0.34	72.45	0.34	72.45	0.34	73.36	0.34
Led	75.66	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.41	0.76	75.56	0.76	75.56	0.76	75.56	0.76
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	84.26	0.84	83.58	0.83
Soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92
Mean	82.88	0.65	82.87	0.65	82.23	0.65	81.82	0.66	81.07	**0.67**	80.58	0.66	77.13	0.65	81.69	0.66

TABLE F.7: AUC values produced when using the two path strategy with respect to a CARM generated DAG model and a range of values for σ

Data set	Threshold Value (σ)											
	$\sigma = 95$		$\sigma = 70$		$\sigma = 60$		$\sigma = 50$		$\sigma = 45$		$\sigma = 40$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Waveform	68.54	0.69	68.54	0.69	68.54	0.69	68.54	0.69	58.92	0.59	55.06	0.55
Wine	86.26	0.86	86.26	0.86	86.26	0.86	86.26	0.86	70.73	0.70	70.73	0.70
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43
Heart	53.76	0.20	53.76	0.20	53.76	0.20	53.76	0.20	42.16	0.18	18.63	0.18
PageBlocks	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20	89.77	0.20
Dermatology	79.62	0.71	79.62	0.71	79.62	0.71	79.62	0.71	79.91	0.74	80.82	0.77
Glass	61.71	0.36	61.71	0.36	61.71	0.36	61.71	0.36	61.71	0.39	57.35	0.38
Zoo	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52
Ecoli	32.42	0.25	32.42	0.25	32.42	0.25	32.42	0.25	39.51	0.28	39.77	0.26
Led	42.06	0.43	42.06	0.43	40.06	0.43	40.06	0.43	39.56	0.41	39.56	0.41
PenDigits	41.62	0.41	41.62	0.41	41.62	0.41	46.76	0.47	32.57	0.33	32.57	0.33
Mean	66.42	0.46	66.42	0.46	66.23	0.46	**66.70**	**0.47**	62.70	0.43	59.92	0.43

Appendix G

Determining the Best Threshold Value With respect to the DAG Classification Model

TABLE G.1: Results obtained using a range of σ values with respect to Naive Bayes classification, the multiple paths strategy and all-level DAGs

Data set	Threshold Value (σ)															
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	56.93	0.29	56.93	0.29	56.93	0.29	56.94	0.29	61.50	0.32	61.50	0.32	61.50	0.32	56.94	0.29
heart	55.22	0.35	55.22	0.35	55.22	0.35	54.88	0.35	54.88	0.35	54.53	0.34	53.91	0.34	54.88	0.35
PageBlocks	91.83	0.53	91.83	0.53	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54
Dermatology	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	86.94	0.85	86.66	0.85	87.23	0.85
glass	69.81	0.46	69.81	0.46	70.29	0.46	72.51	0.50	71.16	0.50	70.69	0.50	68.78	0.50	72.99	0.51
Zoo	92.18	0.58	92.18	0.58	92.18	0.58	92.18	0.58	93.18	0.59	93.18	0.59	93.18	0.59	92.18	0.58
ecoli	84.43	0.41	84.69	0.41	84.08	0.40	82.87	0.38	82.26	0.38	82.26	0.38	82.26	0.38	82.56	0.38
led	75.66	0.76	75.63	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76
PenDigits	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83	83.58	0.83
soybean	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92	90.75	0.92
Mean	78.76	0.60	78.79	0.60	78.77	0.60	78.84	0.60	**79.20**	**0.60**	79.09	0.60	78.81	0.60	78.85	0.60

TABLE G.2: Results obtained using a range of σ values with respect to Naive Bayes classification, the multiple path strategy and two-level DAGs

Data set	Threshold Value (σ)																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.0		0.5×10^{-4}		0.7×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	58.03	0.30	58.03	0.30	58.03	0.30	59.10	0.32	59.11	0.33	59.11	0.33	58.03	0.30	59.14	0.33	59.11	0.33
Heart	54.19	0.35	54.19	0.35	54.19	0.35	54.19	0.35	54.19	0.35	53.84	0.34	54.19	0.35	54.19	0.35	54.19	0.35
PageBlocks	91.83	0.53	91.83	0.53	91.83	0.53	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54
Dermatology	86.66	0.85	86.66	0.85	86.66	0.85	86.66	0.85	86.37	0.84	86.37	0.84	86.66	0.85	86.66	0.85	86.66	0.85
Glass	59.49	0.49	59.49	0.49	59.49	0.49	57.58	0.48	57.58	0.48	57.58	0.48	58.06	0.49	57.58	0.48	57.58	0.48
Zoo	94.18	0.61	94.18	0.61	94.18	0.61	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59
Ecoli	80.23	0.37	80.23	0.37	80.23	0.37	79.93	0.37	79.93	0.37	79.93	0.37	79.93	0.37	79.93	0.37	79.93	0.37
Led	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75	75.56	0.75
PenDigits	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84	83.62	0.84
Soybean	90.39	0.92	90.39	0.92	90.39	0.92	90.39	0.92	90.39	0.92	90.22	0.92	90.39	0.92	90.39	0.92	90.39	0.92
ChessKRvK	18.62	0.32	18.62	0.32	18.62	0.32	18.7	0.32	18.97	0.32	18.97	0.32	18.64	0.32	18.79	0.32	18.83	0.32
LetterRecog	55.71	0.56	55.71	0.56	55.71	0.56	55.70	0.56	55.70	0.56	55.70	0.56	55.70	0.56	55.70	0.56	55.70	0.56
Mean	70.71	0.57	70.71	0.57	70.71	0.57	70.54	0.57	70.54	0.57	70.50	0.57	70.49	0.57	70.55	**0.58**	70.55	0.58

TABLE G.3: Results obtained using a range of σ values with respect to CARM, the multiple paths strategy and all-level DAGs

Data set	Threshold Value (σ)							
	$\sigma = 50$		$\sigma = 40$		$\sigma = 45$		$\sigma = 47$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	86.81	0.43	86.81	0.43	86.81	0.43	86.81	0.43
Heart	53.07	0.20	17.32	0.16	42.51	0.18	53.07	0.20
PageBlocks	91.27	0.22	90.90	0.28	91.21	0.22	91.23	0.22
Dermatology	79.62	0.71	81.28	0.78	79.91	0.74	79.62	0.71
Glass	61.71	0.36	57.35	0.38	62.11	0.39	61.71	0.36
Zoo	88.00	0.52	88.00	0.52	88.00	0.52	88.00	0.52
Ecoli	32.42	0.25	40.68	0.27	39.51	0.28	34.20	0.26
Led	42.06	0.43	39.56	0.41	39.56	0.41	40.44	0.42
PenDigits	41.62	0.42	29.21	0.30	32.71	0.33	32.71	0.33
Mean	64.06	0.41	59.01	0.39	62.48	0.39	63.09	0.38

TABLE G.4: Results obtained using a range of σ values with respect to CARM, the multiple paths strategy and two-level DAGs

Data set	Threshold Value (σ)							
	$\sigma = 50$		$\sigma = 40$		$\sigma = 45$		$\sigma = 47$	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	86.81	0.43	62.44	0.45	62.44	0.45	62.44	0.45
heart	51.70	0.20	34.71	0.15	36.09	0.15	33.68	0.16
PageBlocks	91.47	0.45	90.75	0.32	90.75	0.32	90.75	0.32
Dermatology	65.83	0.55	68.48	0.57	68.48	0.57	68.48	0.57
glass	59.81	0.34	39.24	0.34	39.24	0.34	39.24	0.34
Zoo	86.00	0.50	88.00	0.52	88.00	0.52	88.00	0.52
ecoli	62.94	0.23	23.22	0.13	23.22	0.13	23.22	0.13
led	19.28	0.18	20.13	0.19	19.28	0.18	20.13	0.19
PenDigits	18.95	0.19	16.81	0.16	16.81	0.16	16.81	0.16
soybean	90.04	0.92	90.04	0.92	90.04	0.92	90.04	0.92
chessKRvK	63.28	0.40	53.38	0.38	53.44	0.37	53.28	0.38
Mean	63.28	0.40	53.38	0.38	53.44	0.37	53.28	0.38

Appendix H

Determining the Best Threshold Values With respect to the DAG Classification Model with Breadth Pruning

TABLE H.1: Results obtained using a range of α values with respect to Max-levels DAG, using the single-path strategy

Data set	Threshold Value (α)																			
	0.10		0.20		0.30		0.40		0.50		0.60		0.70		0.80		0.90		0.45	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	56.93	0.29	56.93	0.29	56.93	0.29	79.83	0.40	79.83	0.40	79.83	0.40	79.83	0.40	79.83	0.40	79.83	0.40	79.83	0.40
Heart	56.25	0.36	57.01	0.39	57.01	0.36	57.01	0.36	57.01	0.36	57.01	0.36	57.01	0.36	57.01	0.36	57.01	0.36	57.01	0.36
PageBlocks	91.83	0.53	91.83	0.53	92.69	0.52	92.69	0.52	92.69	0.52	92.69	0.52	92.69	0.52	92.69	0.52	92.69	0.52	92.69	0.52
Dermatology	87.23	0.85	87.23	0.85	86.94	0.85	86.94	0.85	86.94	0.85	86.94	0.85	86.94	0.85	86.94	0.85	86.94	0.85	86.94	0.85
Glass	69.81	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46	69.26	0.46
Zoo	92.18	0.58	90.18	0.56	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54
Ecoli	84.43	0.41	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39	83.52	0.39
Led	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76
PenDigits	83.59	0.84	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83
Soybean	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92
Mean	78.85	0.60	78.52	0.60	78.48	0.59	**80.77**	**0.60**	80.77	0.60	80.77	0.60	80.77	0.60	80.77	0.60	80.77	0.60	80.77	0.60

TABLE H.2: Results obtained using a range of α values with respect to Min-levels DAG, using the single-path strategy

Data set	Threshold Value (α)																			
	0.10		0.20		0.30		0.40		0.50		0.60		0.70		0.80		0.45		0.35	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	63.36	0.32	63.03	0.32	63.03	0.32	90.12	0.45	71.49	0.47	71.64	0.48	91.44	0.54	91.44	0.54	71.33	0.32	63.03	0.32
Heart	55.98	0.36	55.35	0.36	55.01	0.37	59.91	0.40	59.29	0.39	59.29	0.39	59.29	0.39	59.29	0.39	58.60	0.37	56.73	0.35
PageBlocks	91.87	0.54	91.87	0.54	91.87	0.54	92.02	0.49	92.02	0.49	92.02	0.49	92.02	0.49	92.02	0.49	92.02	0.49	91.87	0.54
Dermatology	86.09	0.84	86.09	0.84	86.09	0.84	85.80	0.84	85.23	0.82	82.55	0.79	82.55	0.79	82.55	0.79	86.43	0.85	85.80	0.84
Glass	57.58	0.48	57.58	0.48	57.58	0.48	57.10	0.48	55.75	0.48	58.61	0.47	56.38	0.46	56.38	0.46	56.15	0.48	57.10	0.48
Zoo	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	92.18	0.58	90.18	0.58	90.18	0.58	93.18	0.59	93.18	0.59
Ecoli	79.98	0.37	82.40	0.40	82.40	0.40	82.40	0.40	81.45	0.39	80.54	0.37	80.23	0.38	80.23	0.38	81.79	0.39	82.40	0.40
Led	75.75	0.76	75.75	0.76	75.75	0.76	75.72	0.75	74.16	0.74	64.47	0.65	64.47	0.65	64.47	0.65	75.50	0.75	75.75	0.76
PenDigits	83.62	0.84	83.62	0.84	83.62	0.84	83.68	0.84	82.75	0.83	78.92	0.79	78.92	0.79	78.92	0.79	83.84	0.84	83.60	0.84
Soybean	89.68	0.92	89.68	0.92	89.68	0.92	89.86	0.92	90.04	0.92	87.92	0.87	85.58	0.85	85.58	0.85	89.86	0.92	89.68	0.92
ChessKRvK	17.71	0.33	17.71	0.33	17.71	0.33	18.33	0.33	21.37	0.34	30.40	0.34	34.58	0.33	34.58	0.33	19.27	0.33	17.84	0.33
LetterRecog	55.84	0.56	55.84	0.56	55.84	0.56	55.84	0.56	53.31	0.53	49.44	0.49	49.44	0.49	49.44	0.49	55.83	0.56	55.85	0.56
Mean	70.89	0.58	71.01	0.58	70.98	0.58	**73.66**	**0.59**	71.67	0.58	70.67	0.56	72.09	0.56	72.09	0.56	71.98	0.57	71.07	0.58

TABLE H.3: Results obtained using a range of σ values for following multiple paths using the Max-levels DAG variation and $\alpha = 0.40$ with respect to breadth pruning

Data set	Threshold Value (σ)																			
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.0		0.5×10^{-4}		0.5×10^{-5}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	79.83	0.40	79.83	0.40	79.83	0.40	79.85	0.40	82.32	0.41	82.32	0.41	82.32	0.41	82.32	0.41	82.32	0.41	82.32	0.41
Heart	57.01	0.36	57.08	0.36	57.08	0.36	57.08	0.36	56.32	0.36	55.63	0.35	55.63	0.35	55.63	0.35	57.08	0.36	56.67	0.35
PageBlocks	92.69	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.65	0.52	92.64	0.52	92.65	0.52	92.69	0.52
Dermatology	86.94	0.85	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	86.94	0.85	86.94	0.85	86.37	0.85	87.23	0.85	86.94	0.85
Glass	69.26	0.46	71.64	0.50	71.64	0.50	71.64	0.50	68.38	0.50	65.12	0.49	63.70	0.48	63.70	0.48	71.24	0.51	69.81	0.47
Zoo	89.18	0.54	89.18	0.54	89.18	0.54	89.18	0.54	92.18	0.58	93.18	0.59	93.18	0.59	93.18	0.59	91.18	0.56	89.18	0.54
Ecoli	83.52	0.39	82.87	0.39	82.87	0.39	82.87	0.39	82.26	0.39	82.26	0.39	82.26	0.39	82.26	0.39	81.96	0.38	82.87	0.39
Led	75.66	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.56	0.76
PenDigits	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.02	0.83	83.11	0.83	84.32	0.84	83.02	0.83	83.02	0.83
Soybean	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.39	0.92	90.57	0.92	90.57	0.92
Mean	80.768	0.603	80.960	0.607	80.960	0.607	80.962	0.607	81.046	**0.612**	80.722	0.611	80.589	0.610	80.634	0.611	81.278	0.610	80.963	0.604

TABLE H.4: Results obtained using a range of σ values for following multiple paths using the Min-levels DAG variation and $\alpha = 0.40$ with respect to breadth pruning

Data set	Threshold Value (σ)																	
	0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.0		0.5×10^{-4}	
	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	71.33	0.39	71.33	0.39	71.33	0.39	68.79	0.39	66.28	0.39	66.28	0.39	66.28	0.39	66.28	0.39	71.33	0.39
Heart	59.91	0.40	59.91	0.40	59.91	0.40	59.36	0.40	59.64	0.40	59.29	0.40	59.64	0.40	58.95	0.39	59.91	0.40
PageBlocks	92.02	0.49	92.02	0.49	91.87	0.46	92.02	0.47	92.05	0.47	92.07	0.48	92.09	0.48	92.07	0.48	92.02	0.49
Dermatology	85.80	0.84	85.80	0.84	86.09	0.84	86.66	0.85	86.66	0.85	86.37	0.85	86.37	0.85	86.09	0.85	85.80	0.84
Glass	57.10	0.48	57.10	0.48	56.63	0.48	56.70	0.49	56.23	0.49	56.23	0.49	56.23	0.49	56.23	0.49	57.10	0.48
Zoo	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59
Ecoli	82.40	0.40	82.10	0.40	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	82.40	0.40
Led	75.72	0.75	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.56	0.76	75.72	0.75
PenDigits	83.68	0.84	83.68	0.84	83.68	0.84	83.68	0.84	83.68	0.84	83.68	0.84	83.72	0.84	83.96	0.84	83.68	0.84
Soybean	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92	89.86	0.92
ChessKRvK	18.33	0.33	18.33	0.33	18.34	0.33	18.34	0.33	19.25	0.34	19.24	0.34	19.24	0.34	19.24	0.34	18.33	0.33
LetterRecog	55.84	0.56	55.84	0.56	55.84	0.56	55.84	0.56	55.84	0.56	55.90	0.56	55.98	0.56	56.03	0.56	55.84	0.56
Mean	72.098	0.583	72.059	0.583	71.932	0.580	71.740	0.583	71.593	0.583	71.546	0.584	71.587	**0.584**	71.528	0.583	72.098	0.583

TABLE H.5: Results obtained using a range of σ values for following multiple paths within the DAG using the Max-levels DAG variation and the best α value for each dataset with respect to breadth pruning

Data set	auc	Threshold Value (σ)													
		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.0	
		ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	0.40	79.83	0.40	79.83	0.40	79.85	0.40	82.32	0.41	82.32	0.41	82.32	0.41	82.32	0.41
Heart	0.20	57.01	0.39	57.01	0.39	56.32	0.38	56.25	0.37	55.22	0.37	54.94	0.35	54.60	0.34
PageBlocks	0.20	91.83	0.53	91.85	0.53	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54	91.87	0.54
Dermatology	0.20	87.23	0.85	87.23	0.85	87.23	0.85	87.23	0.85	86.94	0.85	86.94	0.85	86.66	0.85
Glass	0.10	69.81	0.46	70.29	0.46	72.51	0.50	71.16	0.50	70.21	0.50	68.30	0.50	68.30	0.50
Zoo	0.10	92.18	0.58	92.18	0.58	92.18	0.59	92.18	0.59	92.18	0.59	92.18	0.59	92.18	0.59
Ecoli	0.10	84.43	0.41	84.08	0.40	82.87	0.38	82.26	0.38	82.26	0.38	82.26	0.38	82.26	0.38
Led	0.40	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76	75.53	0.76
PenDigits	0.10	83.59	0.84	83.59	0.84	83.59	0.84	83.59	0.84	83.59	0.84	83.59	0.84	84.34	0.84
Soybean	0.40	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.57	0.92	90.39	0.92
Mean		81.201	0.614	81.216	0.613	81.252	0.616	81.296	0.616	81.069	0.616	80.850	0.614	80.845	0.613

TABLE H.6: Results obtained using a range of σ values for following multiple paths using the Min-levels DAG variation and the best α value for each dataset with respect to breadth pruning

Data set	auc	Threshold Value (σ)																	
		0.1×10^{0}		0.1×10^{-1}		0.1×10^{-2}		0.1×10^{-3}		0.1×10^{-4}		0.1×10^{-5}		0.1×10^{-6}		0.0		0.5×10^{-4}	
		ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC	ACC.	AUC
Nursery	0.70	91.44	0.54	91.44	0.54	91.44	0.54	91.44	0.54	90.02	0.58	86.57	0.57	80.46	0.54	78.22	0.53	91.44	0.54
Heart	0.40	59.91	0.40	59.91	0.40	59.91	0.40	59.36	0.40	59.64	0.40	59.29	0.40	59.64	0.40	58.95	0.39	59.36	0.40
PageBlocks	0.40	92.02	0.49	92.02	0.49	91.87	0.46	92.02	0.47	92.05	0.47	92.07	0.48	92.09	0.48	92.07	0.48	92.02	0.47
Dermatology	0.30	86.09	0.84	86.09	0.84	86.09	0.84	85.51	0.84	85.51	0.84	85.23	0.84	85.23	0.84	85.23	0.84	85.51	0.84
Glass	0.30	57.58	0.48	57.58	0.48	57.58	0.48	57.66	0.49	57.18	0.49	57.18	0.49	57.18	0.49	57.18	0.49	57.66	0.49
Zoo	0.40	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59	93.18	0.59
Ecoli	0.40	82.40	0.40	82.10	0.40	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39	80.89	0.39
Led	0.30	75.75	0.76	75.72	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76	75.66	0.76
PenDigits	0.45	83.84	0.84	83.84	0.84	83.84	0.84	83.84	0.84	83.84	0.84	83.84	0.84	83.87	0.84	84.05	0.84	83.84	0.84
Soybean	0.50	90.04	0.92	90.04	0.92	90.04	0.92	90.04	0.92	90.04	0.92	90.04	0.92	90.04	0.92	90.22	0.92	90.04	0.92
ChessKRvK	0.70	34.58	0.33	34.58	0.33	34.58	0.33	34.70	0.34	35.36	0.36	35.42	0.36	35.42	0.36	35.42	0.36	34.81	0.34
LetterRecog	0.35	55.85	0.56	55.85	0.56	55.84	0.56	55.84	0.56	55.84	0.56	55.90	0.56	55.90	0.56	56.02	0.56	55.86	0.56
Mean		75.223	0.596	75.196	0.596	75.077	0.593	75.012	0.595	74.934	**0.600**	74.606	0.600	74.130	0.598	73.924	0.596	75.023	0.595